TUGAS AKHIR – TE145561

DESAIN DAN PENGEMBANGAN *GRAPHICAL USER INTERFACE* PADA *INERTIAL NAVIGATION SYSTEM* KENDARAAN SELAM

Amirotul Khoiro
NRP 10311500000086

Pembimbing
Imam Arifin, S.T., M.T.
Mahardhika Pratama, S.T., M.Sc., Ph.D.
M. Fajar Adianto, S.T.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – TE145561

DESIGN AND DEVELOPMENT GRAPHICAL USER INTERFACE ON INERTIAL NAVIGATION SYSTEM OF SUBMARINE

Amirotul Khoiro
NRP 10311500000086

Supervisor
Imam Arifin, S.T., M.T.
Mahardhika Pratama, S.T., M.Sc., Ph.D.
M. Fajar Adianto, S.T.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul:

"DESAIN DAN PENGEMBANGAN *GRAPHICAL USER INTERFACE* PADA *INERTIAL NAVIGATION SYSTEM* KENDARAAN SELAM"

Adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri. Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Juli 2018

Amirotul Khoiro
NRP. 10311500000086

Halaman ini sengaja dikosongkan

DESAIN DAN PENGEMBANGAN *GRAPHICAL USER INTERFACE* PADA *INERTIAL NAVIGATION SYSTEM* KENDARAAN SELAM

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk
Memperoleh Gelar Ahli Madya
Pada
Departemen Teknik Elektro Otomasi
Institut Teknologi Sepuluh Nopember

Menyetujui

Pembimbing I,	Pembimbing II,	Pembimbing III,
Imam Arifin, S.T., M.T.	Mahardhika, P.S.T.,M.Sc.,PhD	M. Fajar Adianto, S.T.
NIP. 197302222002121001		NIK. 020582016

SURABAYA
JULI, 2018

Halaman ini sengaja dikosongkan

DESAIN DAN PENGEMBANGAN *GRAPHICAL USER INTERFACE* PADA *INERTIAL NAVIGATION SYSTEM* KENDARAAN SELAM DI PT BHIMASENA RESEARCH AND DEVELOPMENT

TUGAS AKHIR

Disusun oleh:

Amirotul Khoiro NRP. 10311500000086

Menyetujui,

Kepala *Human Resources Department*,	Pembimbing Perusahaan,
Fadli Tirmissi	M. Fajar Adianto, S.T.
NIK. 020492016	NIK. 020582016

Chief Executive Officer,

BHIMASENA Research & Development

Dipl. -Ing. Aris Budiyarto

Halaman ini sengaja dikosongkan

DESAIN DAN PENGEMBANGAN *GRAPHICAL USER INTERFACE* PADA *INERTIAL NAVIGATION SYSTEM* KENDARAAN SELAM

Amirotul Khoiro
10311500000086

Pembimbing I : Imam Arifin,S.T., M.T.
Pembimbing II : Mahardhika Pratama, ST., M.Sc., Ph.D.
Pembimbing III : M. Fajar Adianto, S.T.

ABSTRAK

KTBA memerlukan INS dan GPS untuk mengetahui *attitude* (sikap), pergerakan, kecepatan, dan posisi. INS dapat memberikan data *attitude* dan pergerakan serta perkiraan kecepatan dan posisi ketika sinyal GPS hilang di dalam air. INS pada KTBA diimplementasikan menggunakan sensor inersia Sublocus yang dikembangkan oleh Advanced Navigation. Untuk mengetahui data navigasi tersebut diperlukan sebuah proses yang dinamakan antarmuka (*interfacing*).

Proses ini membutuhkan sebuah prosesor untuk membaca data dari sensor inersia tersebut. Prosesor yang akan digunakan adalah Raspberry Pi 3 Model B. Selain pemrosesan data, diperlukan sebuah tampilan antarmuka yang menghubungkan pengguna (*user*) dengan proses yang terjadi. *Graphical User Interface* berfungsi sebagai penghubung antamuka sensor sistem navigasi inersia. Desain antarmuka ini dibuat menggunakan pemrograman Python dengan *library* PyQt. Pengujian sistem dilakukan dengan menghitung *framerate* tiap detiknya (*fps*).

Framerate yang dihasilkan pada penggunaan *Personal Computer* yakni mencapai 50.2999 *fps*. Sedangkan pada penggunaan Single Board Computer Raspberry Pi *framerate* yang dihasilkan lebih kecil yakni 23.67513 *fps*. Hasil pengujian memiliki perbedaan *framerate* sebesar 56.83 % antara *Personal Computer* dengan Raspberry Pi.

Kata Kunci: KTBA, INS, Raspberry Pi 3 Model B, *interfacing*, *Graphical User Interface.*

Halaman ini sengaja dikosongkan

DESIGN AND DEVELOPMENT GRAPHICAL USER INTERFACE ON INERTIAL NAVIGATION SYSTEM OF SUBMARINE

Amirotul Khoiro
10311500000086

Supervisor I : Imam Arifin, S.T., M.T.
Supervisor II : Mahardhika Pratama, S.T., M.Sc., Ph.D.
Supervisor III : M. Fajar Adianto, S.T.

ABSTRACT

KTBA needs INS and GPS to know attitude, orientation, , acceleration, velocity, and position. INS can provide attitude and movement data as well as long distance. INS on KTBA is implemented using Sublocus inertial sensor developed by Advanced Navigation. To know the navigation data is needed a process called interfacing. This process requires a processor to read the data from inertial sensors.

Processor to be used is Raspberry Pi 3 Model B. In addition to data warehouse, required a user connected interface (user) with the process that occurred. Therefore, in this final project built a Graphical User Interface for inertial navigation system. This interface design uses PyQt libraries with Python *programming. The GUI testing is done by calculating the frame rate every second.*

Framerate generated on the use of Personal Computer reaches 50.2999 fps. While on the use of Single Board Computer Raspberry Pi framerate produced smaller is 23.67513 fps. Credit results have a framerate value of 56.83% between Personal Computer with Raspberry Pi.

Keywords: *KTBA, INS, Raspberry Pi 3 Model B, interfacing, Graphical User Interface.*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Segala puji dan syukur kehadirat Allah SWT atas segala rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul: "**Desain dan Pengembangan *Graphical User Interface* pada *Inertial Navigation System* Kendaraan Selam"**. Tugas Akhir ini disusun sebagai salah satu syarat kelulusan yang harus ditempuh oleh setiap mahasiswa di Departemen Teknik Elektro Otomasi ITS. Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada berbagai pihak yang telah memberikan bantuan dan dukungan dalam penyelesaian Tugas Akhir, terutama kepada: Abi, Umi, dan keluarga yang senantiasa memberikan semangat dan doa yang tidak terbatas. Bapak Ir. Joko Susila., M.T., selaku Kepala Departemen Teknik Elektro Otomasi ITS yang selalu mendukung program magang di industri dan Bapak Imam Arifin, S.T, M.T. selaku dosen pembimbing sekaligus Kepala Laboratorium yang telah banyak memberikan bimbingan, saran, serta masukan yang sangat berarti bagi penulis. Pak Mahardhika Pratama, S.T., M.Sc., Ph.D. selaku dosen pembimbing kedua yang senantiasa membimbing dalam pengerjaan Tugas Akhir ini. Seluruh dosen Departemen Teknik Elektro Otomasi ITS yang telah banyak memberikan ilmu selama penulis menempuh kuliah. Bapak Dipl,-Ing. Aris Budiyarto selaku CEO PT Bhimasena Research and Development yang telah memberi kesempatan program magang dan Pak Fadli selaku HRD yang selalu membantu kami dalam administrasi. Mas Fajar Adianto dan Mas Irfan Septiana Putra selaku pembimbing di PT Bhimasena Research and Development yang telah memberi banyak ilmunya, membimbing, serta mengajarkan hal baru dan juga Mas Luqman Harris selaku Kepala Divisi Tim Underwater-KTBA yang selalu memberi masukan dan arahan. Teman-Teman magang Riza, Livian, Tata, Ardi dan Varisa yang telah berjuang bersama selama magang. Teman-teman di jurusan Teknik Elektro Otomasi ITS yang tidak dapat penulis sebutkan satu persatu, terutama rekan-rekan Anggota Laboratorium ACSL yang selalu memberikan semangat dalam menyelesaikan Tugas Akhir ini. Besar harapan penulis bahwa Tugas Akhir ini dapat memberikan informasi dan manfaat bagi pembaca pada umumnya dan mahasiswa Departemen Teknik Elektro Otomasi ITS.

Surabaya, Juli 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

PT Bhimasena Research and Development adalah perusahaan yang bergerak di bidang riset teknologi pertahanan dan keamanan Indonesia. Perusahaan ini mengembangkan teknologi dan produksi peralatan kendaraan militer baik Angkatan Darat, Angkatan Laut, dan Angkatan Udara untuk mendukung misi pertahanan dan keamanan. Berdirinya perusahaan ini dikarenakan kabutuhan sebagai tempat atau lokasinya kendaraan-kendaraan militer di produksi serta melakukan riset-riset untuk pengembangan peralatan-peralatan militer lainnya. PT Bhimasena Research and Development memproduksi beragam kendaraan dan peralatan untuk memenuhi berbagai kebutuhan militer dan juga misi khusus. Kendaraan yang sedang diproduksi sebagai berikut, Kitchen Truck merupakan kendaraan yang didesain untuk kebutuhandapur lapangan sehingga mampu bermanufer untuk memberikan dukungan logistic bagi seluruh prajurit, Jihandak merupakan kendaraan yang mampu meningkatkan kemananan, efisiensi, dan efektivitas pasukan penjinak dalam merespon ancaman-ancaman strategis bahan peledak, Dekontaminasi merupakan kendaraan yang didesain untuk kebutuhan pelaksanaan kegiatan dekontaminasi nubika (nuklir, biologi, dan kimia) bagi personel materil dan unit kendaraan kecil, Shop Contact merupakan kendaraan yang didesain untuk kebutuhan reaksi cepat tanggap darurat dlaam menanggulangi bantuan alat perbengkelan maupun pengawasan perawatan kendaraan, dan SWG R-1 UAV merupakan unit pesawat tanpa awak yang dapat diterbangkan tanpa menggunakan area landasan baik untuk landing ataupun take-off dikendalikan oleh pilot. Untuk penelitian saat ini meliputi UAV, Rhino robot, *ground vehicle*, dan *underwater vehicle*. Saat ini, PT Bhimasena Research and Development sedang mengembangkan proyek *underwater vehicle* berupa kendaraan selam yang dinamakan KTBA (Kapal Tempur Bawah Air) dan *Diver Propulsion Vehicle* (DPV). Kedua proyek ini memiliki kesamaan yakni keduanya memiliki INS (*Inertial Navigation System*) yang digunakan penentu arah kendaraan saat berada di dalam laut.

KTBA memerlukan INS dan GPS untuk mengetahui *attitude*, pergerakan, kecepatan, dan posisi. Navigasi kapal merupakan bagian terpenting pada kapal untuk dapat mencapai tujuan[17]. INS dapat

memberikan data *attitude* dan pergerakan serta perkiraan kecepatan dan posisi ketika sinyal GPS hilang di dalam air. Prinsip dasar INS adalah pengukuran percepatan benda. Percepatan ini kemudian diintegrasikan terhadap waktu menjadi kecepatan. Kecepatan tersebut kemudian diintegrasikan terhadap waktu menjadi posisi[13]. INS pada KTBA diimplementasikan menggunakan sensor inersia Sublocus yang dikembangkan oleh Advanced Navigation. Navigasi kapal merupakan bagian terpenting pada kapal untuk dapat mencapai tujuan[17].

Untuk membaca data dari sensor tersebut, diperlukan sebuah prosesor sebagai alat antarmuka. Prosesor yang akan digunakan adalah Raspberry Pi 3 Model B. Raspberry Pi adalah komputer dalam satu *singleboard* berukuran kecil dengan kapabilitas komunikasi periferal yang luas[18]. Selain antarmuka komunikasi USB (*Universal Serial Bus*), Raspberry Pi memiliki pin GPIO (*General-Purpose Input/Output*) yang mendukung protokol komunikasi I2C (*Inter-Integrated Circuit*) dan protokol SPI (*Serial Peripheral Interface*).

Selain pemrosesan data, didesain sebuah *interface* untuk memudahkan pengawasan yang dapat mengintegrasikan sistem sensor inersia pada kendaraan selam. Perangkat lunak *interface* dikembangkan untuk berinteraksi dengan pengguna dan GUI (*Graphical User Interface*) yang diimplemetasikan dalam *software* PyCharm digunakan untuk menampilkan data *orientation, position, velocity, acceleration, depth, g force, filter status, system status, framerate dll.* Pembuatan GUI pada pemrograman Python ini menggunakan *library* PyQt4.

Sebagai salah satu upaya Angkatan Laut Indonesia dalam mempertahankan ketahanan negara, belakangan ini KTBA Kopaska Angkatan Laut sering digunakan tugasnya dalam sabotase bawah air terhadap instalasi musuh[32]. Kendaraan ini berfungsi sebagai alat pendukung misi pertahanan, keamanan pengintaian, dan SAR (*Search and Rescue*) bawah air. KTBA berfungsi meminimalkan faktor kesulitan yang diemban personil Kopaska saat menyelam dan berenang dengan jarak jauh menyusup ke daerah lawan dengan beban peralatan tempur yang cukup berat[32].

1.2 Rumusan Masalah

Sensor yang digunakan oleh INS pada KTBA adalah sensor inersia Sublocus yang dikembangkan oleh Advanced Navigation. Sensor ini mengeluarkan banyak paket data yang tidak semuanya dibutuhkan. Data pada Sublocus terdiri dari beberapa paket ANPP (*Advanced Navigation*

Packet Protocol). Untuk mengetahui data-data tersebut, maka dibutuhkan sebuah tampilan berupa *Graphical User Interface* yang dapat memudahkan pengguna (*user*) selama berada di dalam KTBA.

1.3 Batasan Masalah

Penelitian ini menggunakan sensor inersia Sublocus yang dikembangkan oleh Advanced Navigation. GUI dibuat untuk *engineer* sebagai media pengujian sensor Sublocus. Komunikasi yang digunakan adalah komunikasi serial dengan panjang baudrate 115200. *Graphical User Interface* dirancang menggunakan *Personal Computer* dengan GUI operasi Windows 10 64-bit. Pengujian dilakukan di tempat darat. Pengujian dilakukan dengan menggunakan *Personal Computer* dan Raspberry Pi. Pengujian GUI dilakukan dengan menganalisa *framerate* yang dihasilkan. Data yang ditampilkan sebanyak 19 data dari ANPP Packet 20 dan 28. Data yang diuji meliputi data *Roll, Pitch,* dan *Yaw*.

1.4 Tujuan

Tujuan utama penelitian ini adalah pembuatan desain GUI untuk membaca dan menampilkan data yang dibutuhkan oleh KTBA dari sensor inersia Sublocus melalui Raspberry Pi.

1.5 Metodologi

Metodologi yang digunakan terdiri dari empat tahap. Tahap pertama melakukan studi literatur, tahap selanjutnya perancangan *interface*, kemudian pengujian dan penyusunan buku. Pada tahap studi literatur, dilakukan pengumpulan dasar teori yang menunjang penguasaan terhadap permasalahan yang berhubungan dengan *interfacing* (antarmuka) sensor inersia menggunakan prosesor. Studi literatur berupa rujukan dalam bentuk jurnal internasional maupun nasional, buku, dan artikel di internet, maupun dari Tugas Akhir mahasiswa sebelumnya.

Tahap selanjutnya meliputi perancangan perangkat keras dan lunak. Pada perancangan perangkat keras dilakukan penyusunan komponen yang terdiri dari rangkaian sensor inersia Sublocus, antena GPS Poseidon, prosesor Raspberry Pi, dan *underwater monitor*. Hasil pembacaan data selanjutnya akan ditampilkan pada GUI. Sedangkan pada tahap perancangan perangkat lunak, dibuat program dari Python untuk mengakuisisi data dari sensor menggunakan prosesor Raspberry

Pi 3 Model B serta membuat GUI menggunakan *software* JetBrains PyCharm Community Edition 2017 2.3.

Setelah melakukan perancangan *interface*, maka dilakukan tahap pengujian sistem dan analisisnya. Pengujian dilakukan untuk mengetahui kinerja pembacaan sensor untuk kemudian mengintegrasikannya pada GUI yang telah dibuat. Data hasil pembacaan selanjutnya dibandingkan dengan data yang menggunakan metode lain.

Penyusunan buku berupa laporan ilmiah yang mencakup seluruh proses pengerjaan Tugas Akhir. Buku ini terdiri dari dasar teori yang dipelajari, analisis dari data yang didapatkan, hingga kesimpulan dan saran.

1.6 Sistematika Penulisan

Untuk pembahasan yang lebih rinci, buku Tugas Akhir ini disusun dengan sistematika penulisan yang terdiri dari lima bab. Bab 1 adalah pendahuluan. Bab ini berisi tentang uraian latar belakang, rumusan masalah , batasan masalah, tujuan penelitian, metodologi penelitian, dan sistematika penulisan, dari penelitian yang dilakukan.

Bab 2 *Inertial Navigation System* pada Kendaraan Selam, bab ini membahas tentang teori dasar yang menunjang, antara lain teori tentang komponen penyusun antarmuka sensor inersia pada kendaran selam seperti sensor inersia Sublocus, Raspberry Pi 3 Model B, antena GPS Poseidon, pemrograman Python, komunikasi serial RS422, dan *underwater monitor*.

Selanjutnya, Bab 3 Perancangan *Graphical User Interface*. Bab ini membahas langkah-langkah perancangan dan pembuatan *interface* secara keseluruhan yang terdiri dari perancangan perangkat lunak. Perancangan perangkat lunak meliputi program *serial reading*, *Graphical User Interface*, perhitungan *framerate*, dan *data processing*.

Bab 4 pengujian dan analisis. Bab ini berisi tentang penjelasan mengenai data dari hasil pengujian beserta analisis dari data yang didapatkan tersebut. Kemudian terakhir, Bab 5 penutup. Bab ini berisi kesimpulan dari proses perancangan dan pengujian sistem antarmuka beserta analisisnya yang telah dilakukan pada bab sebelumnya. Selain itu, bab ini juga berisi saran yang dapat diberikan untuk penelitian selanjutnya.

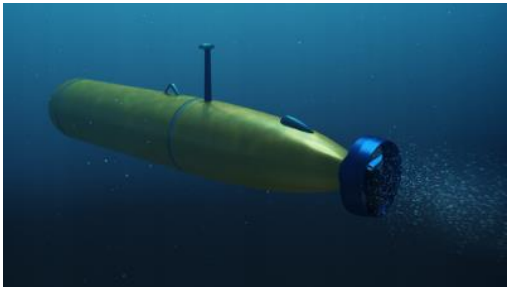
BAB II
INERTIAL NAVIGATION SYSTEM KENDARAAN SELAM

2.1 Kendaraan Selam

Kendaraan yang dapat ditenggelamkan membantu meningkatkan kemampuan manusia untuk menyelidiki lautan dalam. Sebagai contoh, *Manned Submercible* seperti ALVIN yang membawa pengendara melakukan pengamatan langsung dari dalam laut. Pengendara dapat bereaksi terhadap situasi yang mengejutkan dan baru, mereka dapat mengubah eksperimen dan merancang yang baru dengan cepat. Namun, kendaraan selam ini tidak dapat bertahan lama di bawah air. Kemudian *Automated Underwater Vehicles,* kendaraaan ini menggabungkan unsur robotic, *thrusters*, sonar, dn sensor dengan *artificial intelligence.* Beroperasi terus menerus tanpa kendali manusia. Sementara sebagian besar digunakan eksperimen, kendaraan ini sangat menjanjikan untuk ekplorasi laut dalam karena dapat melakukan tugas rutin di bawah air selama berbulan-bulan dalam satu waktu. Juga terdapat *Remotely-Operated Vehicles* seperti JASON yakni kendaraan yang dioperasikan dari jarak jauh yang dikendalikan manusia yang menjelajahi kedalaman dari jarak aman, biasanya kapal induk berada di permukaan laut. ROV dilengkapi dengan lengan manipulator, kamera, lampu, dan sensor untuk melihat dan merasakan untuk pengendaranya. ROV dapat tinggal lebih lama dari kendaraan yang membawa orang dan dapat mengirimkan data secara konstan.

Bhimasena Research & Development sedang mengembangkan sebuah proyek wahana laut yang berupa kendaraan bawah laut dan dinamakan Kapal Tempur Bawah Air (KTBA). Kapal ini berfungsi sebagai alat pendukung misi pertahanan, keamanan, pengintaian, dan SAR bawah air. Kendaraan selam ini berjenis *Swimmer Delivery Vehicle* (SDV). Kendaraan selam diluncurkan dari induk kapal yang dapat diangkut pada *trailer* beban berat[10]. KTBA Kopaska Angkatan Laut sering digunakan tugasnya dalam sabotase bawah air terhadap instalasi musuh. KTBA berfungsi meminimalkan faktor kesulitan yang diemban personil Kopaska saat menyelam dan berenang dengan jarak jauh menyusup ke daerah lawan dengan beban peralatan tempur yang cukup berat[32].

Gambar 2.1 adalah SDV berada di bawah permukaan laut, lalu terendam lebih dalam untuk menghindari deteksi radar musuh, SDV berlabuh di posisi dasar laut yang aman untuk melepaskan perenang dan menyelesaikan misi. SDV dapat melakukan tugas-tugas seperti pengawasan sisi laut, penetralan, misi penyelamatan, pengintaian terselubung di pesisir.



Gambar 2. 1 Kendaraan Selam

Pada kapal selam, berat keseluruhan dari kapal harus sama dengan berat dari air yang dipindahkan. Karena isi dari air yang dipindahkan saat di permukaan kurang daripada saat menyelam, maka diperlukan *ballast* dalam bentuk air yang dihisap ke dalam kapal selam untuk menenggelamkan kapal ini dari permukaan. Pada saat di permukaan, menyelam ataupun berlayar di dalam air, stabilitas statis dari kapal selam haruslah baik untuk mencegah kapal tidak terbalik saat beroperasi[27].

2.2 *Inertial Navigation System* (INS)

Navigasi inersia merupakan bagian penting dalam menentukan posisi kendaraan. Metode navigasi ini banyak digunakan untuk robot, pesawat terbang, kapal selam, dan pesawat luar angkasa. Tujuan utama sistem navigasi adalah untuk mengikuti posisi objek secara terus menerus [29]. INS menyediakan pengukuran dengan akselometer dan giroskop untuk melacak posisi dan orientasi suatu objek relatif terhadap titik awal, orientasi, dan kecepatan. Sebagai *dead-reckoning* navigasi, INS membutuhkan penyelarasan untuk menentukan inisial kondisi sebelum beroperasi. Kecepatan awal dan posisi dapat dengan mudah

diperoleh dengan menggunakan referensi eksternal sistem navigasi seperti *Global Positioning System* (GPS) [31]. *Dead-reckoning* adalah proses memperkirakan posisi sistem dengan menggunakan posisi sebelumnya (koordinat yang disediakan oleh sistem GPS) dan kemudian menghitung posisi baru berdasar gerakan yang dirasakan dari sistem [24]. INS dibagi menjadi dua kategori yakni: *platforms*-INS dan *strapdown*-INS [22]. INS memberikan informasi posisi yang akurat untuk periode waktu singkat sekalipun. Eror terakumulasi seiring berjalannya waktu. Akumulasi ini mengarah ke sebuah kesalahan posisi yang sangat besar membutuhkan sensor tambahan yang diperlukan untuk mengimbangi kesalahan posisi INS [13].

2.2.1 Sensor Inersia Sublocus

Sublocus adalah sistem navigasi inersia bawah air yang memberikan posisi, kecepatan dan orientasi yang akurat pada kedalaman hingga 3000 meter. Sensor ini dikembangkan oleh Advanced Navigation yang merupakan perusahaan Australia yang berfokus pada teknologi dan navigasi bawah air. Sublocus memiliki fitur giroskop-serat-optik pencari arah utara akurasi tinggi, akselerometer, penerima GPS internal dan sensor kedalaman tekanan [13]. Bentuk fisik dari sensor inersia Sublocus dapat dilihat pada Gambar 2.2.

Sublocus adalah navigasi kendaraan bawah air otonom[14]. Sensor ini memiliki akurasi tinggi dan keamanan membuatnya ideal untuk digunakan kontrol juga menyediakan kemampuan untuk melakukan pengaturan sistem, penghindaran rintangan dan kontrol titik otonom sepenuhnya.

Gambar 2. 2 Sensor Inersia Sublocus

Sublocus adalah referensi sensor yang baik untuk kendaraan penelitian bawah laut. Dukungan periferal yang luas dan menjamin untuk menambahkan dukungan untuk perangkat baru berarti bahwa teknologi sensor eksperimental dapat diintegrasikan dalam solusi Sublocus. Spesifikasi dapat dilihat pada Lampiran B1-B-7. Sensor ini dapat memberikan akurasi posisi bawah air di 0,08%[13].Komunikasi ke unit Sublocus melalui antarmuka utama (COM1) ada dalam *Advanced Navigation Packet Protocol* (ANPP). Format RS232 atau RS422 ditetapkan pada 1 bit awal, 8 bit data, 1 *stop* bit dan tidak ada paritas.

ANPP adalah protokol biner yang dirancang dengan pemeriksaan kesalahan yang tinggi, efisiensi tinggi dan praktik desain yang aman. Protokol memiliki spesifikasi yang terdefinisi dengan baik dan sangat fleksibel. Tipe data yang digunakan dalam protokol paket adalah semua tipe data dalam protokol. Header LRC (*Longitudinal Redundancy Check*) menyediakan pengecekan eror pada *header* paket. Ini juga memungkinkan decoder untuk menemukan awal paket dengan memindai LRC yang valid. LRC dapat ditentukan pada Persamaan (2.1).

$$LRC = ((packet\_id + packet\_length + crc[0] + crc[1]) \wedge 0xFF) + 1 \quad (2.1)$$

Paket ID digunakan untuk membedakan isi paket. ID paket berkisar dari 0 hingga 255. Dalam rentang ini ada tiga sub-rentang yang berbeda, ini adalah paket sistem, paket keadaan dan paket-paket konfigurasi. Paket sistem memiliki ID paket dalam rentang 0 hingga 19. Paket-paket ini diimplementasikan yang sama oleh setiap perangkat menggunakan ANPP. Panjang paket menunjukkan panjang data paket, yaitu dari indeks byte 5 dan seterusnya. Panjang paket memiliki kisaran 0 - 255. CRC adalah CRC16-CCITT. Nilai awalnya adalah 0xFFFF. CRC hanya mencakup data paket.

2.2.2 Antena GPS Poseidon

Sublocus adalah sistem navigasi yang terintegrasi penuh dan memiliki sensor kedalaman dan penerima GPS yang terintegrasi. Sensor inersia ini dilengkapi dengan antena GPS dengan kedalaman 6000 meter. Penerima GPS menyediakan posisi awal pra-penyelaman yang akurat [14].

Antena GPS bawah laut dirancang untuk digunakan pada kendaraan bawah laut yang membutuhkan kemampuan untuk mendapatkan GNSS *fix* saat muncul di permukaan laut [11]. Antena ini

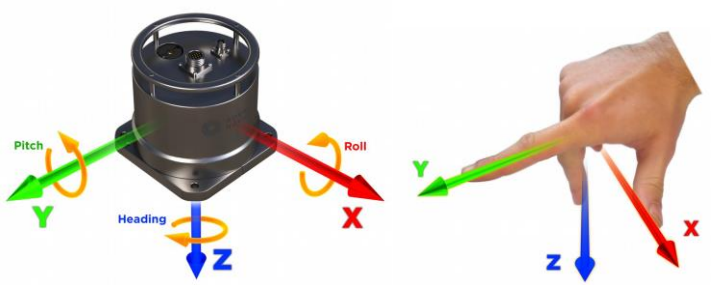
juga cocok untuk kapal laut itu terkena kondisi keras yang terlalu ekstrim untuk antena GPS normal. Antena mampu melacak GPS L1 / L2 / L5, GLONASS G1 / G2 / G3, BeiDou B1 / B2, Galileo E1 / E5 plus Lband. Antena ini ringan, tahan korosi dan mampu tahan terhadap kedalaman sampai 3000 meter. Sistem navigasi permukaan yang telah berhasil dikembangkan dengan mengintegrasikan *Global Positioning System* (GPS) dengan sensor inersia. Dapat dilihat pada Gambar 2.3.

Gambar 2. 3 Antena GPS Poseidon

GPS adalah bagian dari sistem navigasi berbasis satelit yang dikembangkan oleh Departemen Pertahanan AS dibawah satelit NAVSTAR [20]. GPS digunakan untuk penentuan posisi dan waktu yang tepat, menggunakan sinyal radio dari satelit secara *realtime* atau dalam mode pasca-pemrosesan. Sistem ini digunakan di seluruh dunia untuk berbagai navigasi dan aplikasi pemosisian, termasuk navigasi di darat, udara, dan laut seperti navigasi pesawat dan kapal, survei, jaringan kontrol geodetik, studi deformasi kerak, layanan waktu, dll [23].

2.3 Koordinat Sensor

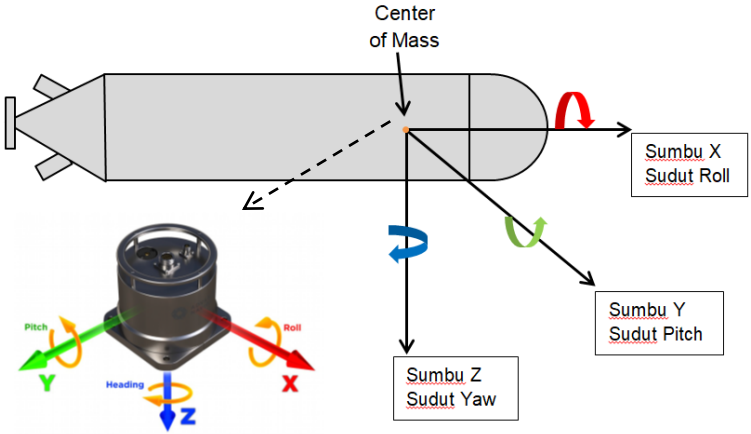
Sensor inersia memiliki 3 sumbu yang berbeda: X, Y dan Z dan ini menentukan arah di mana sudut dan percepatan diukur[13]. Sangat penting untuk menyelaraskan kapal dengan benar dalam pemasangan, jika tidak, sistem tidak akan berfungsi dengan benar. Sumbu-sumbu ini ditunjukkan dalam Gambar 2.4 dengan sumbu X menunjuk ke depan, sumbu Z menunjuk ke bawah dan sumbu Y menunjuk ke kanan. Cara yang baik untuk mengingat sumbu sensor adalah aturan tangan kanan, yang divisualisasikan dalam Gambar 2.4.



Gambar 2. 4 Sumbu Sublocus

2.4 *Roll, Pitch* dan *Yaw*

Orientasi dapat digambarkan oleh tiga sudut *roll*, *pitch* dan *yaw*, ini dikenal sebagai sudut Euler. Sumbu rotasi *roll*, *pitch* dan *yaw* ditampilkan secara visual dalam Gambar 2.5. Panah menunjukkan arah rotasi positif.



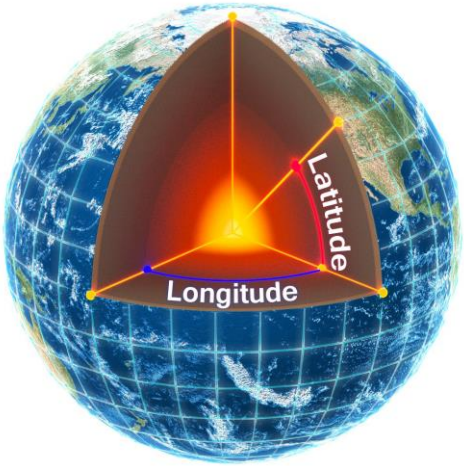
Gambar 2. 5 Orientasi KTBA

Roll adalah sudut di sekitar sumbu X dan nol ketika unit itu rata. *Pitch* adalah sudut di sekitar sumbu Y dan nol ketika unit itu rata. *Heading* adalah sudut di sekitar sumbu Z dan nol ketika sumbu X positif

menunjuk ke utara yang sebenarnya. Posisi INS berada pada titik depan awak kendaraan yaitu tepat di tampat pilot KTBA, tempat ini dinamakan *center of mass* yaitu pusat pergerakan kendaraan.

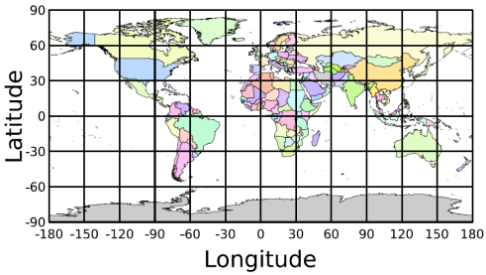
2.5 Koordinat Geodetik

Sistem koordinat geodetik adalah cara paling populer untuk menggambarkan posisi absolut di bumi[13]. Terdiri dari garis lintang dan bujur yang dikombinasikan dengan tinggi relatif terhadap ellipsoid. Garis lintang adalah sudut yang menentukan posisi utara ke selatan dari sebuah titik di permukaan bumi. Bujur adalah sudut yang menentukan posisi timur ke barat dari titik di permukaan Bumi. Garis garis lintang nol adalah khatulistiwa dan garis bujur nol.



Gambar 2. 6 *Longitude* dan *Latitude*

Gambar 2.6 menunjukkan bagaimana sudut lintang dan bujur digunakan untuk menggambarkan posisi di permukaan bumi. *Longitude* dan *latitude* memberi titik 2D di permukaan Bumi. Ini dikombinasikan dengan ketinggian untuk memberikan posisi 3D di bumi. Referensi ellipsoid WGS84 adalah model yang digunakan untuk memperkirakan permukaan laut di seluruh Bumi.

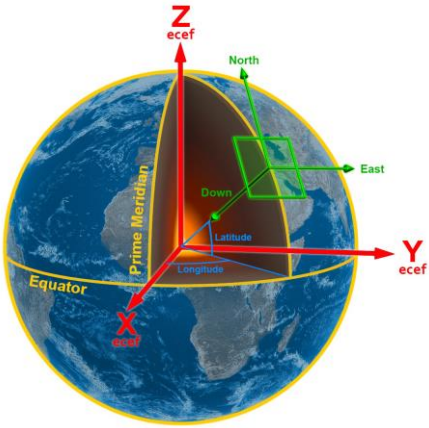


Gambar 2. 7 Peta Dunia

Oleh karena itu ketinggian harus dianggap relatif terhadap permukaan laut. Karena sifat perkiraan model WGS84, ketinggian WGS84 tidak akan sama dengan permukaan laut sebenarnya. Misalnya, di Australia, ketinggian WGS84 di permukaan laut adalah 9 meter di beberapa titik.

2.6 Koordinat NED

Bingkai koordinasi NED (*North East Down*) digunakan untuk mengekspresikan kecepatan dan posisi relatif[13]. Asal-usul bingkai koordinat dapat dianggap sebagai posisi saat ini. Dari asal itu, sumbu utara menunjuk ke utara dan sejajar dengan garis lintang pada titik tersebut.



Gambar 2. 8 Koordinat NED

. Titik sumbu timur tegak lurus dengan sumbu utara dan sejajar dengan garis bujur pada titik itu. Sumbu ke bawah mengarah langsung ke bawah menuju pusat Bumi. Lihat Gambar 2.8 untuk representasi grafis dari kerangka koordinat NED pada posisi di Bumi.

2.7 Koordinat ECEF

Bingkai koordinat ECEF (*Earth-Centred Earth-Fixed*) adalah bingkai koordinat cartesian yang digunakan untuk mewakili posisi absolut di bumi. Asal-usulnya ada di pusat bumi. ECEF adalah alternatif dari kerangka koordinat geodetik. Koordinat ini diwakili oleh tiga sumbu X, Y dan Z yang disajikan secara grafis dalam Gambar 2.8. Posisi ECEF dapat diambil dari produk Navigasi Lanjutan namun sistem geodetik digunakan sebagai *default*.

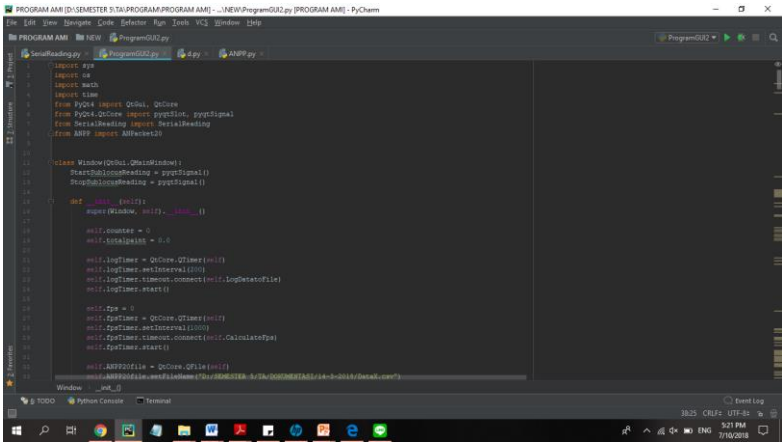
2.8 Pemrograman Python

Python adalah interpreter, berorientasi objek, bahasa pemrograman tingkat tinggi dengan semantic dinamis [13]. Pemrograman disini bertujuan untuk membuat *Graphical User Interface.* GUI adalah antarmuka pada sistem operasi atau komputer yang menggunakan menu grafis. Untuk mengetahui perbedaan pada GUI dapat dianalisa dengan menggunakan prosesor yang digunakan, resoludi layar, *framerate*, penyimpanan memori, dll. Pada penelitian ini parameter yang digunakan untuk menganalisa hasil GUI adalah *framerate.* Pengertian *framerate* adalah jumlah bingkai yang dihasilkan dalam satu detik. Berbeda dengan GUI, HMI (*Human Machine Interface*) adalah antarmuka antara mesin dengan pengguna. Sedikit sulit untuk membedakan manakah antarmuka yang disebut dengan GUI ataupun HMI. Namun pada penelitian ini antarmuka yang dibuat berupa tampilan GUI.

Sintaks sederhana Python mudah dipelajari dan memudahkan pembacaan. Python mendukung berbagai modul dan paket yang mendorong modularitas program. Interpreter Python dan *library* dalam bentuk biner untuk semua platform dan dapat didistribusikan secara bebas. Dalam Tugas Akhir ini, pada perancangan perangkat lunak menggunakan dua *library* yakni PySerial dan PyQt4. PySerial digunakan untuk membaca data dari serial port dan PyQt4 digunakan untuk membuat *Graphical User Interface* (GUI).

2.8.1 *Software* PyCharm

PyCharm adalah *Integrated Development Environment* (IDE) yang digunakan dalam pemrograman komputer yang dikembangkan oleh perusahaan Ceko JetBrains.



Gambar 2. 9 Tampilan *Software* PyCharm

PyCharm menyediakan analisis kode, debugger grafis, unit penguji terpadu, integrasi dengan sistem kontrol versi dan mendukung pengembangan.

2.8.2 PyQt4

Qt adalah *framework* pengembangan aplikasi *cross-platform* yang komperhensif dengan bahasa C++ dan menawarkan solusi pemrograman dengan konsep pemrograman berorientasi objek/OOP (*Object Oriented Programming*) dimana masalah pemrograman diselesaikan dalam objek dengan atribut, fungsi, dan interaksi antar objek [30]. PyQt digunakan secara komersial untuk membuat aplikasi yang ukurannya bervariasi ratusan baris kode lebih dari 100.000 baris kode [26].

Qt mengimplementasikan API (*Application Programming Interface*) secara mandiri diatas API bawaan sistem operasi, sehingga performa dan kapabilitas pada masing-masing platform bekerja secara optimal[30]. Berikut *library* Qt yang digunakan dalam pembuatan GUI:

a. *QLabel*

Kelas ini digunakan untuk menampilkan teks atau gambar. Tidak ada fungsi interaksi pengguna yang disediakan. Tampilan visual label dapat dikonfigurasi dengan berbagai cara dan dapat digunakan untuk menentukan *widget* lain [5].

b. *QLineEdit*

Kelas ini adalah editor teks satu baris. *QLineEdit* memungkinan pengguna untuk memasukkan dan mengedit satu baris teks biasa dengan koleksi fungsi pengeditan yang berguna, termasuk fungsi *undo, redo, cut, paste, drag, and drop* [6].

c. *QCheckBox*

Kelas yang menyediakan kotak centang dengan label teks. *QCheckBox* adalah tombol pilihan yang dapat diaktifkan (dicentang) atau dimatikan (tidak dicentang). Kotak centang biasanya digunakan untuk mewakili fitur dalam aplikasi yang dapat diaktifkan atau dinonaktifkan tanpa mempengaruhi orang lain, namun jenis perilaku yang berbeda dapat diterapkan [1].

d. *QPainter*

Kelas yang dapat menampilkan *draw* tingkat rendah pada *widget* dan perangkat *brush* lainnya. *QPainter* menyediakan fungsi yang sangat optimal untuk melakukan sebagian besar program GUI. *QPainter* bisa menarik semua dari garis sederhana ke bentuk kompleks seperti pie dan akord. Hal ini juga dapat menarik teks dan *pixmaps* yang sejajar. *QPainter* dapat beroperasi pada objek yang mewarisi kelas *QPaintDevice* [7].

e. *QColor*

Kelas *QColor* memberikan warna berdasarkan nilai RGB, HSV, atau CMYK. Warna biasanya ditentukan dalam komponen RGB (red, green, blue), namun juga memungkinkan untuk menentukan dalam bentuk HSV (*hue, saturation, value*), dan CMYK (*cyan, magenta, yellow, dan black*) komponen. Selain itu warna bisa ditentukan dengan menggunakan nama warna [2].

f. *QFile*

Kelas *QFile* menyediakan antarmuka untuk membaca dan menulis file. *QFile* adalah perangkat I/O untuk membaca dan menulis teks dan file biner. *QFile* dapat digunakan dengan sendirinya atau lebih mudah dengan *QTextStream* atau *QDataStream*. Nama file biasanya dilewatkan ke konstruktor, tapi bisa diatur menggunakan *setFileName*(). *QFile*

mengharapkan pemisah file menjadi '/' terlepas dari sistem operasi. Penggunaan pemisan (mis, '\') tidak didukung [3].

g. *QIODevice*

Kelas *QIODevice* adalah kelas antarmuka dasar semua perangkat I/O di Qt. *QIODevice* menyediakan implementasi umum dan antarmuka abstrak untuk perangkat yang mendukung pembacaan dan penulisan blok data, seperti *QFile*, *QBuffer* dan *QTopSocket*. *QIODevice* bersifat abstrak dan tidak dapat di inisialisasikan, namun biasanya menggunakan antarmuka yang didefinisikan untuk menyediakan fitur perangkat mandiri I/O. Sebagai contoh, kelas XML Qt beroperasi pada pointer *QIODevice* yang memungkinkannya digunakan dengan berbagai perangkat (seperti *file* dan *buffer*) [4] .

h. *QTimer*

Kelas *QTimer* menyediakan *timer* berulang dan *single-shot*. Kelas *QTimer* menyediakan antarmuka pemrograman tingkat tinggi untuk penghitung waktu. Untuk menggunakannya, buat *QTimer*, hubungkan sinyal *timeout* () nya ke slot yang sesuai, dan panggil mulai (). Sejak saat itu akan memancarkan sinyal *timeout* () pada interval konstan [8].

2.9 Komunikasi Serial

Komunikasi serial ialah pengiriman data secara serial (data dikirim satu persatu secara berurutan), sehingga komunikasi serial lebih lambat daripada komunikasi paralel. Komunikasi serial dapat digunakan untuk menggantikan komunikasi paralel jalur data 8-bit dengan baik [30]. Komunikasi data serial mengenal dua buah metode, yaitu sinkron dan asinkron. Metode sinkron mengirimkan datanya beberapa *byte* atau karakter (atau disebut blok data atau *frame*) sebelum meminta konfirmasi apakah data sudah diterima dengan baik atau tidak. Sementara metode asinkron data dikirim satu *byte* setiap pengiriman. Pada mode asinkron, setiap karakter ditempatkan berada diantara bit *start* dan bit *stop*. Bit *start* selalu satu bit, tapi *stop* bit bisa satu bit atau dua bit. *Start* bit selalu 0 (*low*) dan *stop* bit selalu 1 (*high*).

Komunikasi ke unit Sublocus melalui antarmuka utama (COM1) ada dalam ANPP. Format RS232 atau RS422 ditetapkan pada 1 bit awal, 8 bit data, 1 stop bit dan tidak ada paritas. Baud rate *default* Sublocus adalah 115200. Baud rate dapat diatur di mana saja dari 1200 hingga 10.000.000 baud dan dapat dimodifikasi menggunakan perangkat lunak Sublocus Manager. Penting untuk memilih baud rate yang mampu membawa jumlah data yang diatur Sublocus untuk dikirim.

Dalam telekomunikasi, RS422 adalah nama standar teknis yang menetapkan karakteristik listrik dari rangkaian sinyal digital. Komunikasi ini adalah standar untuk komunikasi serial biner antar perangkat. RS422 adalah protokol atau spesifikasi yang harus diikuti untuk memungkinkan dua perangkat yang menerapkan standar ini untuk berkomunikasi satu sama lain. RS-422 standar adalah versi terbaru dari standar protokol serial asli yang dikenal sebagai RS232. Beberapa sistem terhubung langsung menggunakan sinyal RS422. Dalam komunikasinya, satu perangkat adalah *Data Terminal Equipment* (DTE) dan perangkat lainnya adalah *Data Communication Equipment* (DCE) [12].

Gambar 2.10 adalah RS422 Serial to USB menyediakan serial port DB-9 *male* untuk *device connection*. Transfer datanya hingga 3 Mbps termasuk bus *powered mode*[15]. Untuk spesifikasi lengkapnya dapat dilihat pada Lampiran B-10 dan B-11.

2.10 *Underwater Monitor*

DNC-10V adalah monitor bawah air 10 inci dengan input VGA atau HDMI. Monitor ini memiliki akses menu penuh, memungkinkan pengguna mengakses cepat tanpa menggulir melalui menu. Monitor eksternal yang besar dan cerah memberi kemampuan untuk membingkai gambar dan fokus dengan presisi tinggi dan sesuai dengan hampir semua perumahan, kamera bawah air. Dapat dilihat pada Gambar 2.11. Monitor dengan input VGA bekerja sempurna dengan perangkat, seperti laptop, dan lain-lain. Monitor ini menjamin NEMA 6 (IP 68).

Gambar 2. 10 *Underwater monitor* DNC 10 V

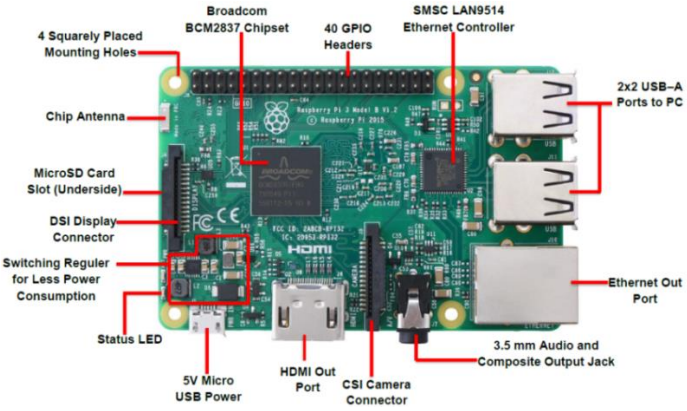
Lapisan aluminium anodized yang kuat tahan terhadap air asin, dirancang untuk menahan lingkungan yang ekstrim [19]. Untuk spesifikasinya dapat dilihat pada Tabel 2.1 lengkapnya pada Lampiran B-9.

Tabel 2. 1 Spesifikasi *Underwater Monitor*

Spesfifikasi	Nilai
Ukuran layar	10”
Rasio tampilan panel	4: 3
Resolusi	1024 x 768 (maks sampai 1920x1440)
HDMI	1080p60; 1080p50; 1080i60; 1080i50
Kedalaman	60 m
Ukuran	232mm x 183mm x 57mm

2.11 *Single Board Computer* Raspberry Pi

Raspberry Pi 3 Model B adalah sebuah komputer papan tunggal yang lengkap dengan mikroprosessor, memori, *input*/*output* dan fitur lain yang dibutuhkan dari sebuah komputer fungsional. Bentuk fisik dari Raspberry Pi 3 dapat dilihat pada Gambar 2.12.



Gambar 2. 11 Raspberry Pi 3 Model B

Pada Tugas Akhir ini prosesor yang digunakan adalah Raspberry Pi 3 Model B. Kapabilitas Raspberry Pi mendekati komputer berukuran

normal, Raspberry Pi juga memiliki pin *General Purpose Input Output* (GPIO) dan protokol komunikasi yang umum seperti I2C (*Inter-Intergrated Circuit*) dan SPI (*Serial Peripheral Interface*), sehingga Raspberry Pi juga memiliki kemampuan yang sama dengan mikrokontroler. Meskipun Raspberry Pi memiliki GPIO, Raspberry Pi hanya mampu menerima data digital.

Hal itu disebabkan karena Raspberry Pi tidak memiliki komponen *Analog-Digital Converter* (ADC). Raspberry Pi dapat menerima data analog secara tidak langsung dengan cara berkomunikasi dengan ADC eksternal yang lain. Spesifikasi dan bentuk fisik dapat dilihat pada Tabel 2.2 dan Gambar 2.12.

Tabel 2. 2 Spesifikasi Raspberry Pi 3 Model B

Spesfifikasi	Nilai
SoC	BCM2837
CPU	Quad Cortex A53 @1.2GHz
Instruction set	ARMv8-A
GPU	400 MHz VideoCore IV
RAM	1 GB SDRAM
Penyimpanan	micro-SD
Ethernet	10/100
Wireless	802.11n/Bluetooth 4.0
Video	HDMI
GPIO	40

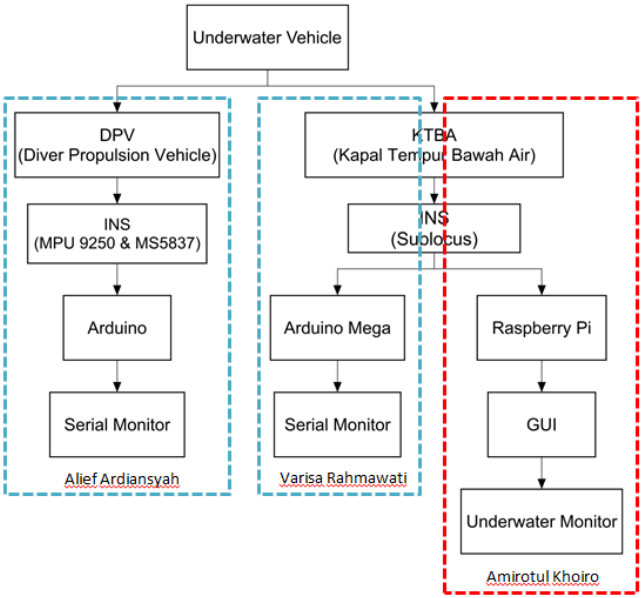
Raspbian adalah sistem operasi yang direkomendasikan untuk penggunaan pada Raspberry Pi [9]. Sistem operasi komputer ini berbasis Debian untuk Raspberry Pi. Ada beberapa versi Raspbian termasuk Raspbian Stretch dan Raspbian Jessie. Sejak tahun 2015 telah secara resmi disediakan oleh Raspberry Pi Foundation sebagai sistem operasi utama untuk keluarga komputer *single board board* Raspberry.

Raspbian diciptakan oleh Mike Thompson dan Peter Green sebagai proyek independen. Pembangunan awal selesai pada bulan Juni 2012. Sistem operasi masih dalam pengembangan aktif. Raspbian sangat dioptimalkan untuk CPU ARM berkinerja rendah Raspberry.

Halaman ini sengaja dikosongkan

BAB III
PERANCANGAN *GRAPHICAL USER INTERFACE*

Pada bab ini dijelaskan mengenai konfigurasi sistem dan perancangan perangkat lunak. Dan juga dijelaskan perancangan pengerjaan INS divisi *Diver Propulsion Vehicle* (DPV) dan Kapal Tempur Bawah Air (KTBA) pada proyek *Underwater Vehicle.*

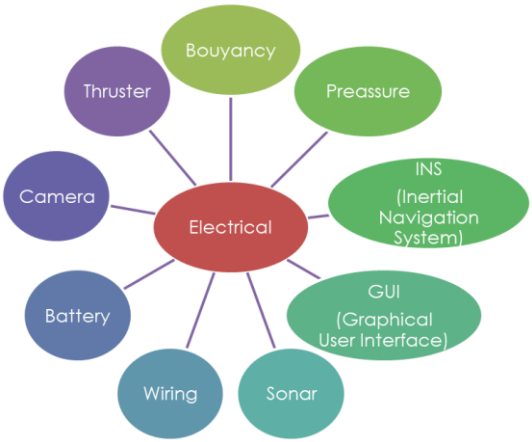


Gambar 3. 1 Diagram Pengerjaan *Underwater Vehicle*

Gambar 3.1 adalah diagram blok konfigurasi pengerjaan penelitian di proyek *underwater vehicle.* Terbagi menjai 2 divisi yakni *Diver Propulsion Vehicle* (DPV) dan Kapal Tempur Bawah Air (KTBA). Penelitian ini berfokus pada bagian *Inertial Navigation System* (INS). Namun, kedua divisi ini memiliki INS yang berbeda. INS yang digunakan pada DPV yaitu sensor inersi MPU9250 dan sensor tekanan MS5837. Sedangkan pada divisi KTBA INS yang digunakan yakni sensor inersia Sublocus yang dikembangkan oleh Advanced Navigation.

Penelitian pada INS DPV ini merancang INS menggunakan sensor tersebut untuk dibandingkan dengan penggunaan sensor analog yang dipakai sebelumnya. Sedangkan penelitian INS KTBA ini mengenai pembacaan data atau biasa disebut dengan parsing data dari sensor dengan menggunakan prosesor Raspberry Pi dan metode lain dengan menggunakan mikrokontroler Arduino Mega. Pada pemrosesan data menggunakan Raspberry Pi ini dirancang sebuah *Graphical User Interface* (GUI) yang ditampilkan pada *underwater monitor.*

Divisi KTBA memiliki bagian-bagian yang dikerjakan *engineer*. Terdapat *electrical*, *mechanical*, dan *material engineer*. Bagian *electrical* pada KTBA dapat dilihat pada Gambar 3.2.

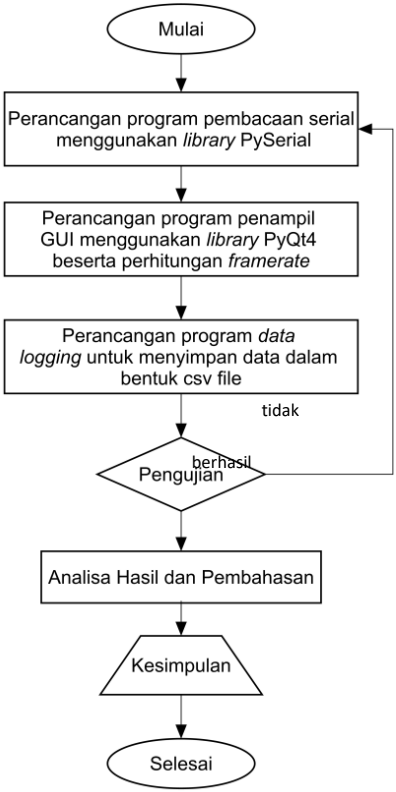


Gambar 3. 2 Bagian *Electrical* KTBA

Bagian *electrical* ini meliputi *buoyancy* atau kemampuan kapal mengapung, *thruster* yaitu baling-baling bagian belakang yang digunakan untuk menggerakkan KTBA. *Camera* yang digunakan adalah Pi NoIR Camera V2 yang terintegrasi dengan prosesor Raspberry Pi. Kamera ini tahan terhadap air, digunakan tanpa *infrared* karena di dalam air tidak dapat menerima banyak cahaya. *Battery* sebagai sumber utama yang dibutuhkan oleh sistem keseluruhan KTBA ini. *Wiring* adalah pengkabelan antara *board controller* dengan komponen-komponen sistem pada KTBA. Sonar digunakan untuk navigasi KTBA

selain menggunakan INS. INS pada KTBA menggunakan sensor Sublocus dan terdapat GUI keseluruhan yang berisi penampilan data sensor inersia Sublocus, sonar, dan *buoyancy*. Pada penelitian kali ini terdapat pembuatan GUI untuk menampilkan data sensor inersia Sublocus. Yang terakhir yakni *preasure* menggunakan sensor tekanan NMEA 183.

Penelitian ini mengenai pemrosesan data dari sensor inersia Sublocus serta penampilan datanya pada *Graphical User Interface* yang ditampilkan pada *underwater monitor*. Berikut rancangan pengerjaan dapat dilihat pada Gambar 3.3.



Gambar 3. 3 Diagram Pengerjaan

Proses pengerjaan ini dimulai dengan perancangan program pembacaan data serial menggunakan *library* PySerial. Perancangan ini membaca data serial dari sensor yang menggunakan komunikasi serial RS422, juga terdapat pembacaan *port serial* RS422 to USB. Setelah merancangan pembacaan serial kemudian pengerjaan perancangan program penampilan GUI menggunakan *library* PyQt4. *Class Qt* yang digunakan adalah *QLabel, QPainter, QCheckBox, QColor, QLineEdit.* Untuk menganalisa GUI maka digunakan *framerate* untuk menghitung banyaknya gambar setiap detiknya. Dirancang juga program perhitungan GUI menggunakan *QTimer*. Perhitungan ini yaitu penjumlahan waktu berakhirnya penampilan GUI dikurangi dengan waktu awal dimulainya penampilan GUI. Setelah perancangan GUI, data yang ditampilkan perlu disimpan. Penyimpanan ini dinamakan *data logging* yaitu proses penyimpanan dalam bentuk file. Penyimpanan data ini berformat csv. Proses penyimpanan ini memanfaatkan *class QFile.* Setelah melakukan berbagai perancangan, dilanjutkan ke pengujian yang meliputi 3 metode penggunaan komputer. Pengujian pertama menggunakan PC, pengujian kedua menggunakan prosesor Raspberry Pi. Sistem ini merupakan sistem yang digunakan pada KTBA, dengan menggunakan Raspberry Pi sebagai prosesor dan ditampilkan pada *underwater monitor.* Pengujian terakhir menggunakan 2 komputer tadi secara bersamaan. Setelah pengujian, didapatkan hasil dan GUI dapat dianalisa dengan membandingkan *framerate* yang dihasilkan. Setelah mendapatkan hasul dan analisa dapat dibuat kesimpulan mengenai penggunaan komputer yang digunakan.

3.1 *Design Requirements*

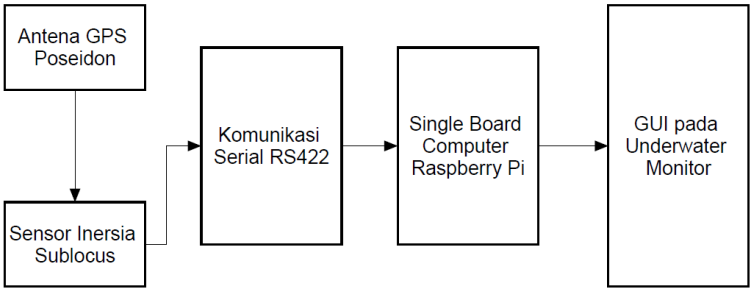
Berdasarkan pembagian pengerjaan pada Gambar 3.1, penulis mengerjakan bagian sensor inersia Sublocus pada Kapal Tempur Bawah Air. Hasil dari penelitian ini adalah sebuah tampilan *Graphical User Interface* (GUI) yang digunakan *user* untuk membaca data dari sensor. Secara garis besar terdapat empat bagian pengerjaan, yakni pembacaan data sensor inersia Sublocus melalui prosesor, penampilan GUI menggunakan *library* PyQt4, perhitungan *framerate* yang dihasilkan oleh GUI, serta penyimpanan data yang telah ditampilkan oleh GUI atau yang disebut dengan *data logging.* Target yang ingin dicapai dalam perancangan GUI ini adalah sebagai berikut:

- Dapat membaca data dari sensor inersia Sublocus yakni pada ANPP *Packet* 20 dan *Packet* 28.

- Dapat menampilkan data ANPP *Packet 20* dan *Packet* 28 sebanyak 19 data.
- GUI yang dibuat dapat beroperasi di sistem operasi Windows pada *Personal Computer* dan Raspbian pada Raspberry Pi.
- GUI yang dibuat mampu menghasilkan *framerate* sebesar 25fps pada Raspberry Pi yang ditampilkan pada *underwater monitor* dan sebesar 50fps pada *Personal Computer*.
- GUI yang dibuat dapat ditampilkan pada layar yang memiliki resolusi sebesar 1366x768 pada *Personal Computer* dan 1280x768 pada *underwater monitor.*

3.2 Konfigurasi Sistem

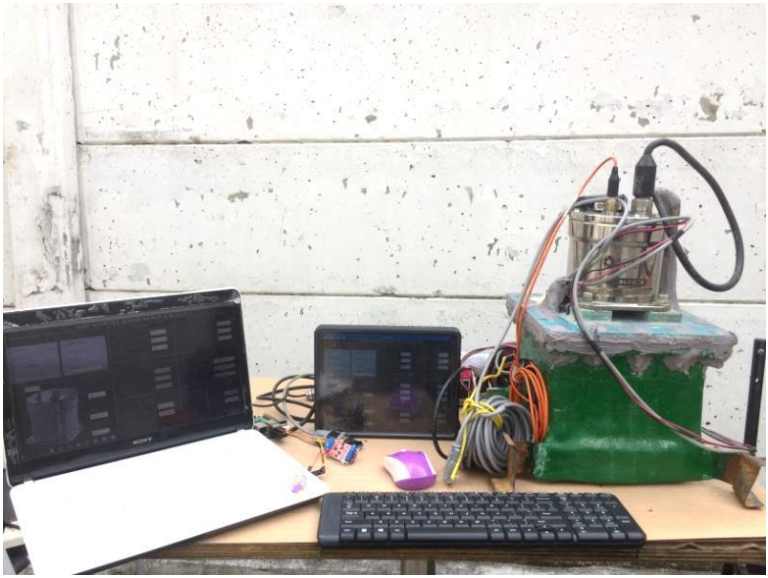
Sensor inersia Sublocus diberi sumber tegangan sebesar 24 V dari baterai. Sublocus dihubungkan dengan antena GPS, yakni subsea Poseidon untuk bisa mengambil data GPS didalam Sublocus.



Gambar 3. 4 Diagram Blok Sistem

Pada Sublocus terdapat kabel utama yang terbagi menjadi empat bagian. Bagian yang dipakai ada primary cable yakni menggunakan komunikasi serial RS422. Kemudian kabel RS422 ini dihubungkan dengan USB to RS422 *converter*, lalu USB dihubungkan ke port USB pada Raspberry Pi. GUI diproses pada Raspberry Pi kemudian ditampikan pada *underwater monitor*.

Sublocus juga dihubungkan dengan kabel RS422 dan dihubungkan ke USB *converter* yakni USB to RS422, kemudian kabel USB masuk ke *port* USB pada Raspberry Pi, dari Raspberry Pi dihubungkan kabel HDMI menuju *underwater monitor*.



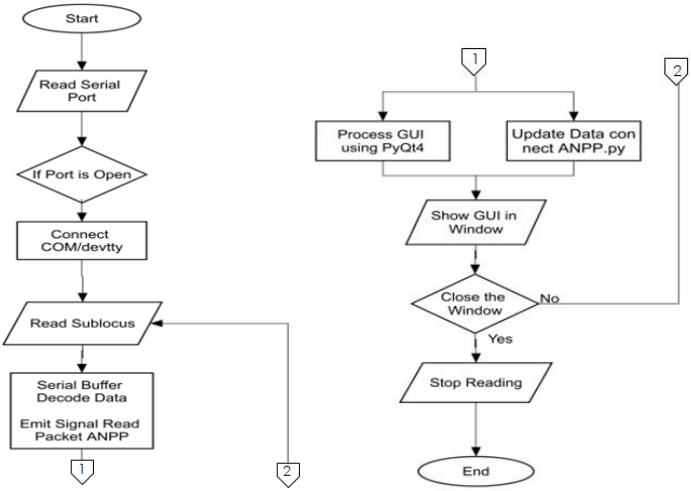
Gambar 3. 5 Sistem Antarmuka pada *Inertial Navigation System*

3.3 Perancangan Perangkat Lunak

Sensor Sublocus mengeluarkan banyak paket data yang tidak semuanya dibutuhkan. Data pada Sublocus terdiri dari beberapa paket ANPP (*Advanced Navigation Packet Protocol*). Untuk mengetahui data-data tersebut, maka dibutuhkan sebuah tampilan berupa *Graphical User Interface* yang dapat memudahkan pengguna (*user*) selama berada di dalam KTBA.

3.3.1 Serial Reading

Program ini berisi pembacaan *port* USB menggunakan *library* PySerial. *Port* USB pada *Personal Computer* bernama COM, sedangkan pada Raspberry Pi bernama devtty. Pada pembacaan serial ini menggunakan *library* PySerial aliran program dapat dilihat pada Gambar 3.6.



Gambar 3. 6 Diagram Alir Program *Serial Reading*

Program dimulai dengan pembacaan *port serial* dilambangkan dengan "*Read Serial Port*". Kemudian jika *port* terbuka maka pembacan ini disambungkan ke *port* COM untuk *port Personal Computer* dan devtty untuk *port* Raspberry Pi dilambangkan dengan "*Connect* COM/devtty". Sensor inersia Sublocus memiliki *baudrate* sebesar 115200 dan *timeout* sebesar 10. Pemilihan *baudrate* disesuaikan dengan data yang ingin ditampilkan dapat dilihat pada Persamaan 3.1.

Data throughput = (112 (packet length) + 5 (fixed packet overhead)) * 50 (rate)
Data throughput = 5850 bytes per second
Minimum baud rate = data throughput x 11 = 64350 Baud
Closest standard baud rate = 115200 Baud

(3.1)

Pembacaan ini terdapat pada program *serial reading* Gambar 3.7. untuk program selengkapnya dapat dilihat pada Lampiran A-1.

```
self.port = serial.Serial(self.portName,
                          baudrate=self.baudrate,
                          parity=serial.PARITY_NONE,
                          stopbits=serial.STOPBITS_ONE,
                          bytesize=serial.EIGHTBITS,
                          writeTimeout=0,
                          timeout=self.timeout,
```

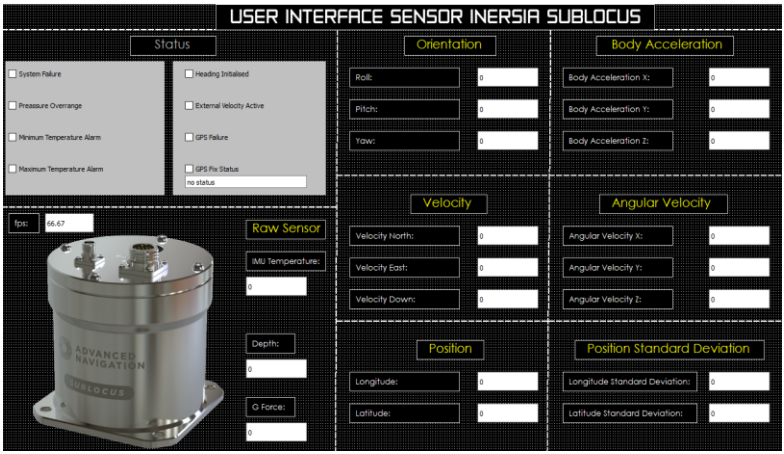
```
                            rtscts=False,
                            dsrdtr=False,
                            xonxoff=False)
self.sublocusWorker = SerialReading("COM7", 115200, 10)
```

Gambar 3. 7 *Listing* Program *Serial Reading*

Ketika USB terdeteksi, data masuk pada *serial buffer*. Pada tahap ini data masuk *buffer* dan ditampung oleh ANPacket Decoder. *Decoder* ini mempunyai ukuran *header* sebesar 5 bit dan maksimal 255 bit, serta ukuran *buffer* sebesar 8 bit. Setelah melalui proses *decoding*, data masuk ke ANPP (*Advanced Navigation Packet Protocol*) milik sensor inersia Sublocus. Paket yang ditampilkan yakni meliputi ANPP Packet ID 20 dengan panjang 112 bit dan ANPP Packet ID 28 dengan panjang 48 bit untuk lebih lengkapnya dapat dilihat pada Lampiran B-12 untuk Tipe data ANPP, B-*13* ANPP *Packet* 20, B-14 ANPP *Packet* 28.

3.3.2 Penampilan GUI dan *Data Processing*

Program ini berisi desain GUI pada pemrograman Python menggunakan library PyQt4. Desain ini memanfaatkan fungsi-fungsi PyQt, sebagian besar menggunakan Qt *QPainter*, *QLineEdit*, dan *QLabel*. Pemrosesan GUI dapat dilihat pada Gambar 3.6.



Gambar 3. 8 Tampilan GUI Menggunakan PyQt4

Pada perancangan ini penggunaan *QLabel* digunakan untuk menampilkan gambar Sublocus. *QLineEdit* digunakan pada bagian penampil data. *QCheckBox* adalah tombol pilihan yang dapat diaktifkan (dicentang) atau dimatikan (tidak dicentang). Pada perancangan ini bagian *QCheckBox* adalah pada menu status. *QPainter* menyediakan fungsi yang sangat optimal untuk melakukan sebagian besar program GUI. Berikut tampilan GUI menggunakan *library* PyQt4. Dapat dilihat pada Gambar 3.8.

Penggunaan *QPainter* terdapat pada semua tulisan, garis, *background*, dll. *QColor* dapat memberikan warna berdasarkan nilai RGB, HSV, atau CMYK. Pada perancangan ini tulisan yang dibuat oleh *QPainter* menggunakan fungsi *QColor*.

QFile menyediakan antarmuka untuk membaca dan menulis file. *QFile* adalah perangkat I/O untuk membaca dan menulis teks dan file biner. Proses penyimpanan file data saat pengujian menggunakan fungsi *QFile* untuk perintah WriteOnly. *QIODevice* adalah kelas antarmuka dasar semua perangkat I/O di Qt. Pada perancangan ini bagian yang menggunakan *QTimer* adalah untuk menghitung *framerate*.

Selain ditampilkan, data juga diproses dilambangkan dengan "*Update data conncet* ANPP.py" dapat dilihat pada Gambar 3.9. Untuk program selengkapnya dapat dilihat pada Lampiran A-2.

```
roll = roll_rad * 180 / math.pi
pitch = pitch_rad * 180 / math.pi
yaw = yaw_rad * 180 / math.pi
latitude = latitude_rad * 180 / math.pi
longitude = longitude_rad * 180 / math.pi
```

Gambar 3. 9 *Listing* program data *orientation* dan *position*

Data *orientation* dan *position* berupa *radian* kemudian diubah menjadi *degree*. Berikut rumus pengubah satuan radian ke *degree*. Dapat dilihat pada Persamaan (3.1).

$$1\,rad = 1 \times 180/\pi \approx 57.2958° \tag{3.1}$$

Selain data *orientation* dan *position*, data *velocity* juga diubah satuannya. Data *velocity* berupa meter per sekon diubah menjadi *knots*. Dapat dilihat pada Gambar 3.11, untuk program selengkapnya dapat dilihat pada Lampiran A-2.

```
velonorth=velonorth_ms*1.9
veloeast=veloeast_ms*1.9
velodown=velodown_ms*1.9
```

Gambar 3. 10 *Listing* program data *velocity*

Digunakan *knots* karena satuan dalam dunia perkapalan lebih mengenal *knots* dibanding meter per sekon. Berikut rumus pengubah satuan meter per sekon ke dalam knots dapat dilihat pada Persamaan (3.2).

$$1\ {}^{m}/_{s}\ \approx 1.9438\ knots \tag{3.2}$$

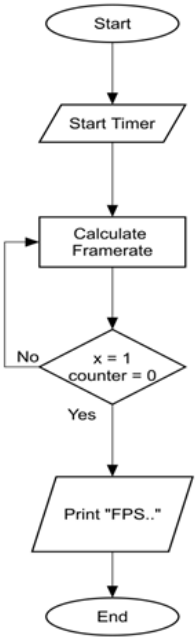
3.3.3 Perhitungan *Framerate*

Framerate pada GUI didapatkan dari penghitungan total waktu ketika berakhirnya pemrosesan GUI dikurangi dengan waktu saat memulainya proses GUI. Digunakan memastikan kecepatan data dari sensor inersia Sublocus dapat dilihat dengan kecepatan *framerate* ketika GUI dijalankan.

```
defpaintEvent(self,event):
    self.counter+=1
    startTime = time.time()
paintduration=time.time()-startTime
self.totalpaint+= paintduration
self.fps=0
self.fpsTimer=QtCore.QTimer(self)
self.fpsTimer.setInterval(1000)
self.fpsTimer.timeout.connect(self.CalculateFps)
        self.fpsTimer.start()
```

Gambar 3. 11 *Listing* Program *Timer*

Gambar 3.11 adalah program dimulai dengan perhitungan waktu menggunakan kelas *QTimer* dilambangkan dengan "*Start Timer*". Dapat dilihat pada Gambar 3.12, untuk program selengkapnya dapat dilihat pada Lampiran A-2. Setelah memulai perhitungan waktu, kemudian menghitung *framerate* yang terdapat pada Gambar 3.12.



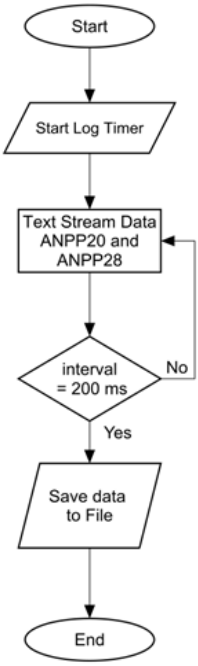
Gambar 3. 12 Diagram Alir Program Perhitungan *Framerate*

Perhitungan fps ini menggunakan kelas *QTimer* pada PyQt4. Setelah dihitung didapatkan fps yang ditampilkan pada GUI. Pada pembuatan awal program *framerate* ini ditampilkan pada *command window* program.

3.3.4 *Data Logging*

Penyimpanan data yang telah di proses dalam bentuk csv file. Perhitungan data dengan interval 200 ms. Proses *logging* data ini memanfaatkan fungsi Qt QFile untuk penyimpanan datanya dan

QTextStream utuk proses streaming data yang masuk dalam file. Dapat dilihat diagram alirnya pada Gambar 3.14.



Gambar 3. 13 Diagram Alir Program *Data Logging*

Program dimulai dengan memlai waktu *logging* dilambangkan dengan "*Start Log Timer*". Data tersimpan sesuai kecepatan interval timer yang telah diatur. Dilambangkan dengan "*interval*=200 ms". Dapat dilihat pada Gambar 3.15, untuk program selengkapnya dapat dilihat pada Lampiran A-2.

```
self.logTimer=QtCore.QTimer(self)
self.logTimer.setInterval(200)
self.logTimer.timeout.connect(self.LogDatatoFile)
self.logTimer.start()
```

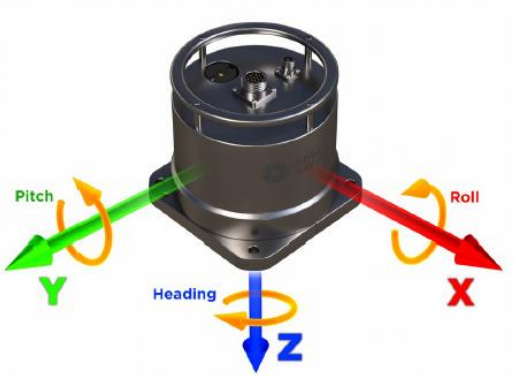
Gambar 3. 14 *Listing* Program *Data Logging*

BAB 1V
PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan pengujian *Graphical User Interface* (GUI) menggunakan tiga metode penggunaan komputer beserta analisisnya. Pengujian meliputi pengujian GUI dengan menggunakan *Personal Computer* (PC), pengujian GUI dengan menggunakan *Single Board Computer* Raspberry Pi, dan pengujian GUI dengan menggunakan PC dan Raspberry Pi secara bersamaaan.

4.1. Kriteria Pengujian

Pengujian GUI menampilkan 19 data dari ANPP Packet 20 dan 28. Data yang diuji yaitu data *orientation* yakni *roll*, *pitch*, dan *yaw* yang sama antara penggunaan dua komputer yang berbeda. Data *orientation* ini dapat terlihat nilainya akibat perlakuan yang dilakukan pada Sublocus. Sublocus diberi perlakuan dengan memutar dan menggerakkan sensor sesuai arah sumbunya dapat dilihat pada Gambar 4.1.



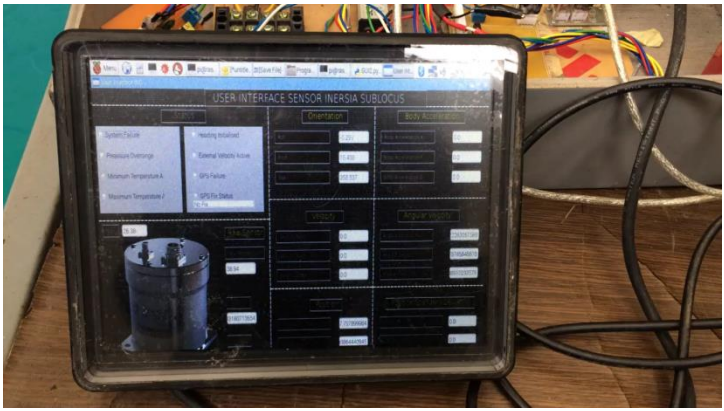
Gambar 4. 1 Pengujian Data Orientation

Untuk mengetahui perbedaan pengujian yang dilakukan, perbedaan GUI dapat dibandingkan dengan melihat *framerate* yang dihasilkan. Berikut spesifikasi komputer beserta CPU *clock* yang dimiliki. Dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Spesifikasi Komputer yang Digunakan

Jenis Komputer	Prosesor	CPU Clock	Resolusi
Personal Computer	Intel core i3	1.8 GHz	1366x768
Raspberry Pi 3 Model B	BCM2837	1.2 GHz	1280x768

Pengujian sensor inersia Sublocua dilakukan di darat karena KTBA sekarang dalam masa penelitian lebih lanjut. Gambar 4.2 merupakan tampilan GUI yang digunakan KTBA. Tampilan ini menggunakan prosesor Raspberry Pi 3 Model B dan ditampilkan pada *underwater monitor*. Digunakan layar yang tahan air karena layar ini digunakan KTBA bersentuhan langsung dengan air saat berada dalam laut.



Gambar 4. 2 Pengujian Tampilan GUI

GUI yang telah ditampilkan diuji kebenaran datanya dengan melakukan pergerakan pada Sublocus. Sensor digerakkan sesuai dengan sumbunya. Gambar 4.3 adalah persiapan pengujian sensor Sublocus. Sebelum melakukan pengujian, sensor inersia Sublocus terlebih dahulu dihubungkan dengan antena GPS Poseidon yang juga milik Advanced Navigation. Digunakan Poseidon untuk membaca GPS supaya mendapatkan data berupa *velocity* dan *acceleration.* Sinyal GPS tidak dapat terbaca saat KTBA berada dibawah laut. Maka dari itu perlu dideteksi terlebih dahulu saat berada di daratan. Pada tampilan GUI juga

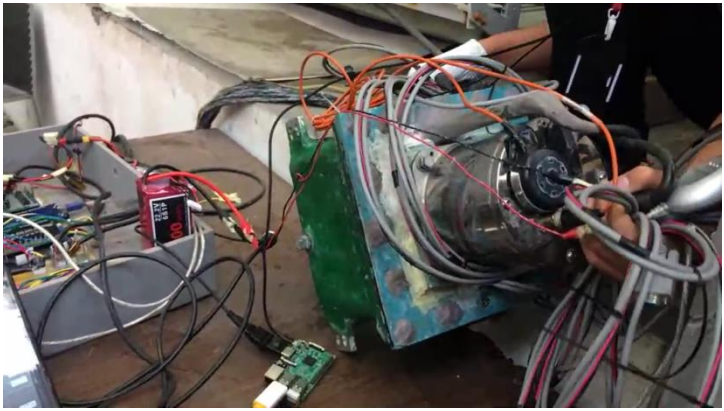
diberi menu status dari GPS yang dapat dideteksi. Apabila yang terdeteksi status "*No Fix*" dan "*2D Fix*" maka pembacaan data belum baik. Akan lebih baik jika GPS yang terdeteksi adalah "*3D Fix*".

Gambar 4. 3 Pengujian Sensor Posisi Awal

Pengujian dengan menggerakkan sensor sesuai dengan sumbunya yaitu XYZ. Gambar 4.4 adalah pengujian sensor yang digerakkan sesuai rotasi sumbu Y, yakni menguji orientasi nilai *Pitch.*

Gambar 4. 4 Pengujian Sensor pada Sumbu Y

Data yang didapatkan pada pengujian Gambar 4. 4 bernilai positif. Sebaliknya, data pada pengujian Gambar 4.5 juga menguji nilai *Pitch* namun data yang dihasilkan beenilai negatif karena berlawanan arah dengan sumbu Y.

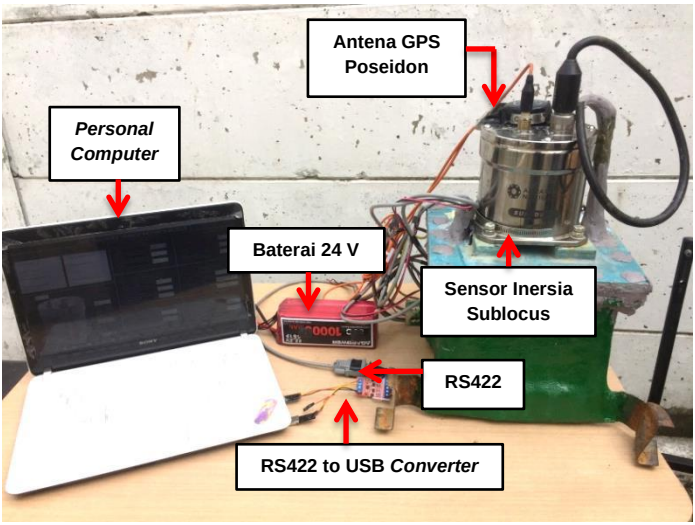


Gambar 4. 5 Pengujian Sensor pada Sumbu -Y

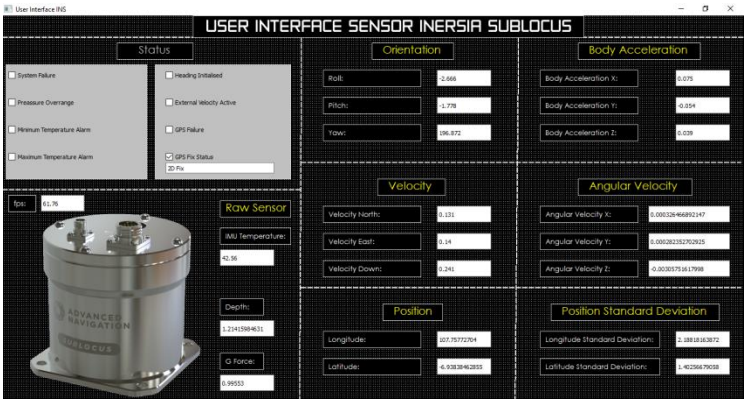
Setelah mengetahui metode pengujian pada sensor, maka pembahasan selanjutnya yaitu pengujian menurut penggunaan komputer. Komputer yang digunakan yaitu *Personal Computer* dan Raspberry Pi. Namun pada sistem KTBA menggunakan Raspberry Pi sebagai prosesor dan penampil GUI. Pengujian menggunakan komputer berbeda ini dengan tujuan membandingkan GUI yang ditampilkan dan dianalisa *framerate*nya sesuai dengan komputer yang digunakan

4.2. Pengujian Menggunakan *Personal Computer*

Pada bagian ini dilakukan pengujian sistem dengan menggunakan *Personal Computer*. Konfigurasi sistem ini berawal dari sensor inersia Sublocus yang diberi tegangan sebesar 24V. Sensor ini dihubungkan dengan antena GPS Poseidon terlebih dahulu. Sensor inersia Sublocus dibaca dengan menggunakan komunikasi serial RS 422. Komunikasi ini memiliki *port* USB yang kemudian disambungkan ke dalam *port* USB pada *Personal Computer*. Jadi pada pengujian ini, prosesor dan juga penampilan GUI tedapat pada *Persnal Computer* sekaligus. Pengujian sistem dan tampilan GUI dapat dilihat pada Gambar 4.6 dan Gambar 4.7.



Gambar 4. 6 Pengujian Menggunakan *Personal Computer*



Gambar 4. 7 Tampilan GUI Pengujian Menggunakan *Personal Computer*

Untuk mengetahui pergerakan nilai data Sublocus maka perlu diberi perlakuan pergerakan sesuai dengan sumbunya. Saat menguji

besarnya nilai *roll*, Sublocus dimiringkan ke kanan dan ke kiri. Saat menguji nilai *pitch*, Sublocus ditundukkan dan ditarik ke arah belakang dari posisi awal. Dan untuk pengujian nilai *yaw*, Sublocus dirotasikan hingga 360 °.

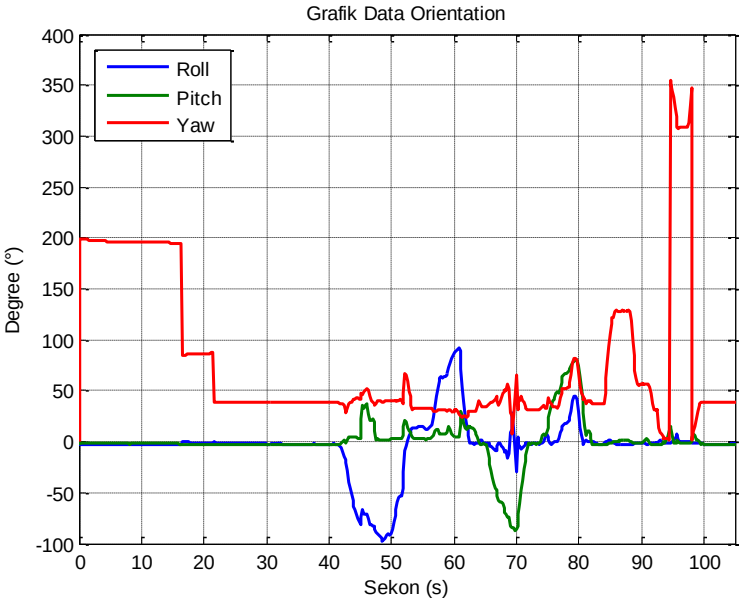
Hasil pengujian sensor inersia Sublocus dengan menggunakan *Personal Computer* ditunjukkan pada Tabel 4.2 dan Gambar 4.4.

Tabel 4. 2 Hasil Pengujian Menggunakan *Personal Computer*

Data	Nilai	
	Maksimum	**Minimum**
Roll	92.3198 °	-97.1572 °
Pitch	80.3265 °	-86.9162 °
Yaw	355.147 °	128.3265 °
Latitude	-6.93829 °	-6.93829 °
Longitude	107.758 °	107.758 °
Velocity North	2.25588 °/s	-3.2309 °/s
Velocity East	5.75798 °/s	-2.77632 °/s
Velocity Down	2.93594 °/s	-2.84366 °/s
Body Acceleration X	2.16585 m/s/s	-2.39055 m/s/s
Body Acceleration Y	1.26799 m/s/s	-1.77517 m/s/s
Body Acceleration Z	1.80947 m/s/s	-1.82664 m/s/s
Angular Velocity X	1.4272 °/s	-1.49628 °/s
Angular Velocity Y	1.11082 °/s	-1.10776 °/s
Angular Velocity Z	2.59022 °/s	-1.70571 °/s
Latitude Standard Deviation	6.67832 m	1.09783 m
Longitude Standard Deviation	7.17591 m	1.77204 m
IMU Temperature	42.84 °	42.55 °
Depth	1.6998 m	0.849898 m
G Force	1.23548 g	0.821219 g

Dari data pengujian tersebut dapat diketahui nilai data dari perbedaan perlakuan Sublocus. Ketika diberi perlakuan, data *roll* menunjukkan angka lebih dari 90° ke kiri maupun ke kanan. Data *pitch* menunjukkan angka 80° hingga -86°. Sedangkan data *yaw* menunjukkan angka hampir 360°.

Untuk mengetahui keakuratan data, nilai pembacaan data dengan perlakuan. Hasil visualisasi dari perlakuan pada Sublocus dapat dilihat pada Gambar 4.8. Dari grafik ini menunjukkan data fluktuasi akibat perlakuan untuk membuktikan data *orientation*.



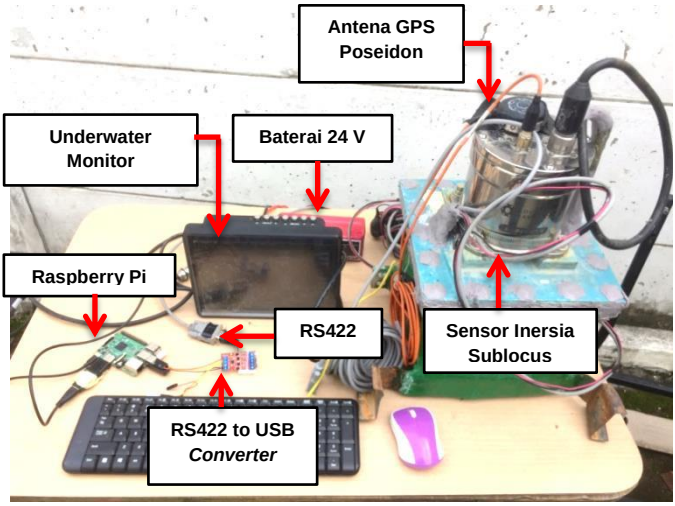
Gambar 4. 8 Grafik Data Pengujian Menggunakan *Personal Computer*

Pada pengujian ini menghasilkan *framerate* dengan rata-rata 62.722 fps. *Framerate* ini dapat dikatakan sangat bagus dengan hasil diatas 50 fps.

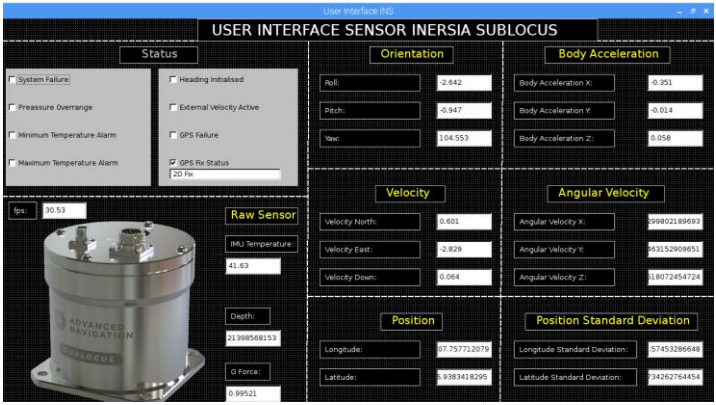
4.3. Pengujian Menggunakan Raspberry Pi

Pada bagian ini dilakukan pengujian sistem dengan menggunakan Raspberry Pi . Konfigurasi sistem ini berawal dari sensor inersia Sublocus yang diberi tegangan sebesar 24V. Sensor ini dihubungkan dengan antena GPS Poseidon terlebih dahulu. Sensor inersia Sublocus dibaca dengan menggunakan komunikasi serial RS 422. Komunikasi ini memiliki *port* USB yang kemudian disambungkan ke dalam *port* USB pada Raspberry Pi dan ditampilkan pada *underwater monitor*. Digunakan *underwater monitor*, karena didalam kendaraan selam air

yang berada diluar juga dapat masuk didalam ruang kendaraan selam ini sehingga membutuhkan layar yang tahan air. Pengujian sistem dan tampilan GUI dapat dilihat pada Gambar 4.9 dan Gambar 4.10.



Gambar 4. 9 Pengujian Sistem Menggunakan Raspberry Pi



Gambar 4. 10 Tampilam GUI Pengujian Menggunakan Raspberry Pi

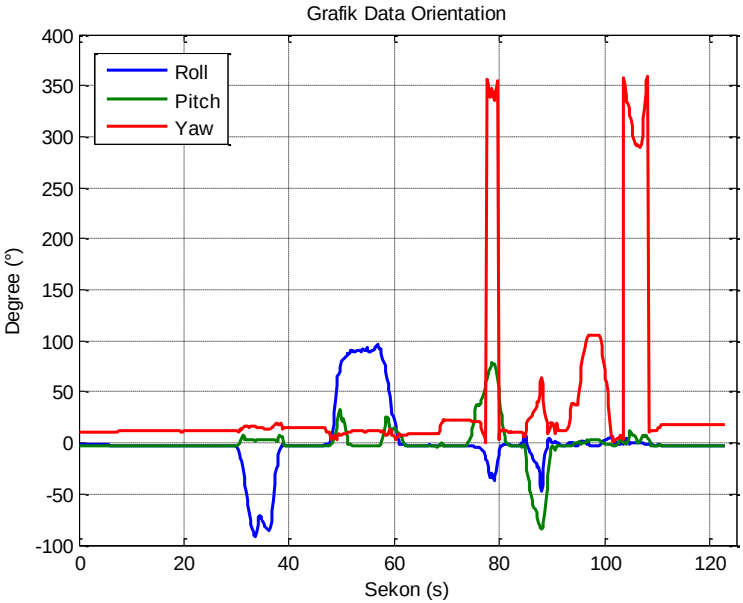
Berikut ini adalah hasil pengujian sensor inersia Sublocus dengan menggunakan Raspberry Pi ditunjukkan pada Tabel 4.4 dan Gambar 4.11.

Tabel 4. 3 Hasil Pengujian Menggunakan Raspberry Pi

Data	Nilai	
	Maksimum	**Minimum**
Roll	96.3416 °	-91.4773 °
Pitch	78.5683 °	-84.153 °
Yaw	357.887 °	105.995 °
Latitude	-6.93856 °	-6.93858 °
Longitude	107.758 °	107.758 °
Velocity North	5.93188 °/s	-0.522522 °/s
Velocity East	0.453232 °/s	-2.15186 °/s
Velocity Down	0.459565 °/s	-1.21605 °/s
Body Acceleration X	1.60283 m/s/s	-1.92453 m/s/s
Body Acceleration Y	1.91648 m/s/s	-3.8495 m/s/s
Body Acceleration Z	1.37076 m/s/s	-1.2172 m/s/s
Angular Velocity X	1.24814 °/s	-0.85693 °/s
Angular Velocity Y	1.30012 °/s	-1.37705 °/s
Angular Velocity Z	1.26077 °/s	-1.26416 °/s
Latitude Standard Deviation	119.178 m	91.0876 m
Longitude Standard Deviation	118.454 m	90.139 m
IMU Temperature	41.13 °	40.66 °
Depth	1.57808 m	0.849736 m
G Force	1.42248 g	0.808999 g

Dari data pengujian tersebut terlihat nilai data dari perbedaan perlakuan Sublocus. Ketika diberi perlakuan, data *roll* menunjukkan angka lebih dari 90° ke kiri maupun ke kanan. Data *pitch* menunjukkan angka 78° hingga -84°. Sedangkan data *yaw* menunjukkan angka hampir 360°.

Untuk mengetahui keakuratan data, nilai pembacaan data dengan perlakuan. Hasil visualisasi dari perlakuan pada Sublocus dapat dilihat pada Gambar 4.11 Dari grafik ini melihatkan data fluktuasi akibat perlakuan untuk membuktikan data orientation. Berbeda dengan metode PC, metode ini memperlihatkan grafik nilai *framerate* yang kecil. Hal ini dikarenakan perbedaan pada penggunaan komputer dan pengambilan data pada waktu yang berbeda.



Gambar 4. 11 Grafik Data Pengujian Menggunakan Raspberry Pi

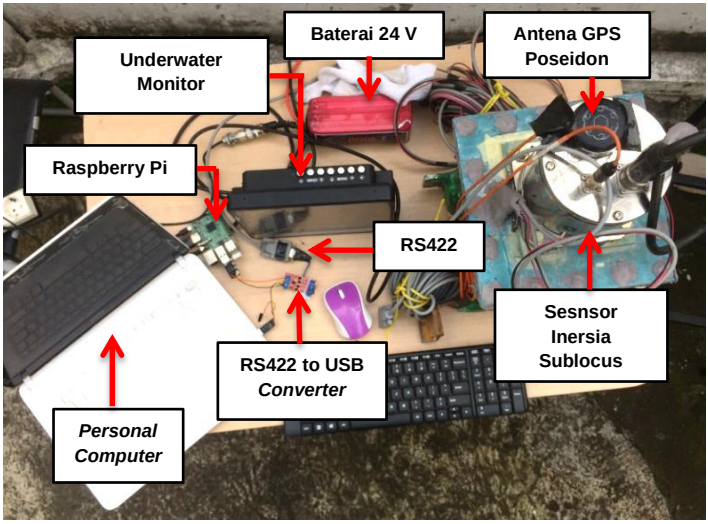
.

Oleh karena ini, pada metode selanjutnya, pengujian dilakukan secara bersamaan dan *realtime* agar dapat mengetahui perbedaan yang signifikan antara penggunaan PC dan Raspberry Pi. Pada pengujian ini menghasilkan *framerate* dengan rata-rata 25.110 fps. *Framerate* ini dapat dikatakan bagus dengan hasil diatas 25 fps. Karena dengan *framerate* sebesar itu sudah dapat dilihat dengan baik.

4.4. Pengujian Menggunakan Dua Komputer Secara Bersamaan

Pada bagian ini dilakukan pengujian sistem keselurahan dengan menggunakan *Personal Computer* dan Raspberry Pi. Konfigurasi sistem ini berawal dari sensor inersia Sublocus yang diberi tegangan sebesar 24V. Sensor ini dihubungkan dengan antena GPS Poseidon terlebih dahulu. Sensor inersia Sublocus dibaca dengan menggunakan komunikasi serial RS 422. Komunikasi ini memiliki *port* USB yang

kemudian disambungkan ke dalam *port* USB pada *Personal Computer* dan Raspberry Pi yang ditampilkan pada *underwater monitor*. Pada pengujian ini *port* USB digunakan dengan 1 *transmitter* dan 2 *receivers*. Berbeda dengan pengujian-pengujian sebelumnya yang hanya menggunakan 1 *transmitter* dan 1 *receiver*.



Gambar 4. 12 Pengujian Menggunakan Dua Komputer

Hal yang membedakan pengujian ini dengan pengujian sebelumnya yakni pada komputer yang digunakan. Pengujian ini, penggunaan dua komputer dilakukan bersamaan dan secara *realtime*. Tujuan dilakukan pengujian bersamaan adalah untuk mengetahui kesamaan data yang ditampilkan dan untuk menemukan perbedaan yang signifikan pada GUI antara kedua komputer. Sehingga dapat Pengujian sistem keseluruhan dapat dilihat pada Gambar 4.12.

Berikut ini adalah hasil pengujian sensor inersia Sublocus dengan menggunakan *Personal Computer* dan Raspberry Pi ditunjukkan pada Tabel 4.6 dan Tabel 4.7 serta Gambar 4.13 dan Gambar 4.12.

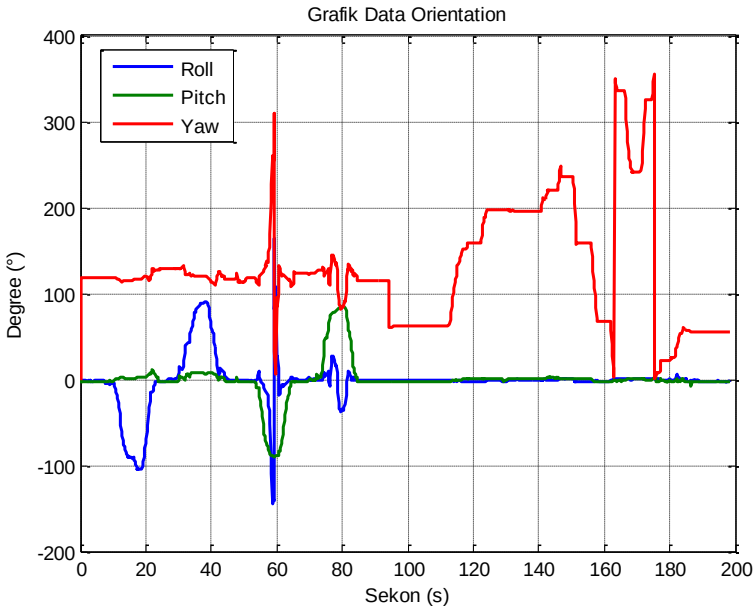
Tabel 4. 4 Hasil Pengujian Menggunakan Dua Komputer (*Personal Computer*)

Data	Nilai	
	Maksimum	**Minimum**
Roll	90.4738 °	-104.798 °
Pitch	85.0704 °	-89.4811 °
Yaw	350.184 °	248.823 °
Latitude	-6.69901°	-6.69903°
Longitude	107.344 °	107.344 °
Velocity North	-3.46399 °/s	-15.1267 °/s
Velocity East	-1.9001 °/s	-6.28156 °/s
Velocity Down	1.14151 °/s	-0.73753 °/s
Body Acceleration X	1.30153 m/s/s	-2.37835 m/s/s
Body Acceleration Y	1.34788 m/s/s	-1.47672 m/s/s
Body Acceleration Z	1.16135 m/s/s	-2.15496 m/s/s
Angular Velocity X	1.19119 °/s	-0.82923 °/s
Angular Velocity Y	1.08321 °/s	-0.95355 °/s
Angular Velocity Z	1.57466 °/s	-1.80649 °/s
Latitude Standard Deviation	49998.9 m	49998.9 m
Longitude Standard Deviation	49998.9 m	49998.9 m
IMU Temperature	39.5 °	38.33 °
Depth	1.57395 m	1.22162 m
G Force	0.847515 g	0.818684 g

Dari data pengujian tersebut terlihat nilai data dari perbedaan perlakuan Sublocus. Ketika diberi perlakuan, data *Roll* menunjukkan angka lebih dari 90° ke kiri maupun ke kanan. Data *Pitch* menunjukkan angka 78° hingga -84°. Sedangkan data *Yaw* menunjukkan angka hampir 360°.

Untuk mengetahui keakuratan data, nilai pembacaan data dengan perlakuan. Hasil visualisasi dari perlakuan pada Sublocus dapat dilihat pada Gambar 4.13. Dari grafik ini melihatkan data fluktuasi akibat perlakuan untuk membuktikan data *orientation*.

Pada pengujian ini menghasilkan *framerate* dengan rata-rata 50.299 fps. *Framerate* ini dapat dikatakan sangat bagus dengan hasil diatas 50 fps. Karena dengan *framerate* sebesar itu sudah dapat dilihat dengan baik.



Gambar 4. 13 Grafik Data Pengujian Menggunakan Dua Komputer (*Personal Computer*)

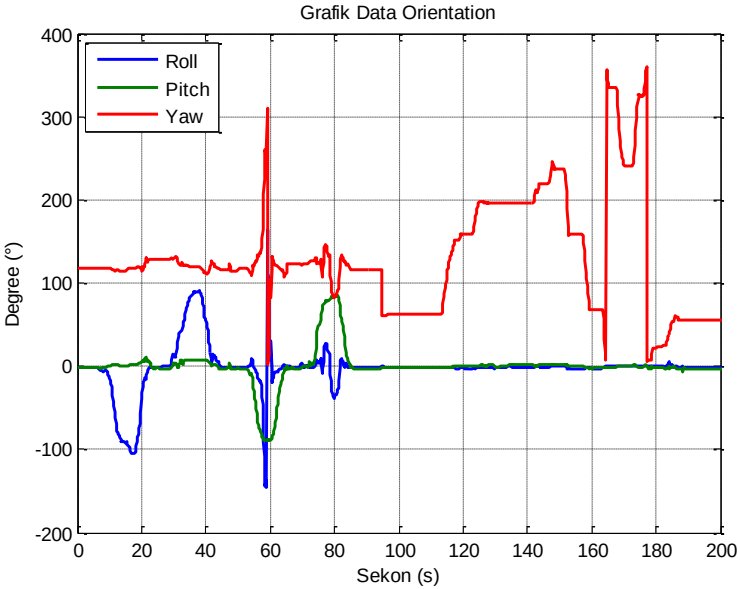
Tabel 4. 5 Hasil Pengujian Menggunakan Dua Komputer (Raspberry Pi)

Data	Nilai	
	Maksimum	**Minimum**
Roll	90.5613 °	-104.798 °
Pitch	85.0704°	-89.6725 °
Yaw	359.327 °	246.102 °
Latitude	-6.69901°	-6.69903°
Longitude	107.344 °	107.344 °
Velocity North	-3.45872 °/s	-15.1443 °/s
Velocity East	-1.9001 °/s	-6.34334 °/s
Velocity Down	1.1456 °/s	-0.73853 °/s
Body Acceleration X	1.30153 m/s/s	-1.81774 m/s/s
Body Acceleration Y	1.12816 m/s/s	-1.47672 m/s/s
Body Acceleration Z	1.32403 m/s/s	-1.00925 m/s/s
Angular Velocity X	1.19119 °/s	-0.82923 °/s
Angular Velocity Y	1.01732 °/s	-0.95355 °/s

Tabel 4. 5 Hasil Pengujian Menggunakan Dua Komputer (Raspberry Pi) (lanjutan)

Data	Nilai	
	Maksimum	**Minimum**
Angular Velocity Z	1.60035 °/s	-1.97159°/s
Latitude Standard Deviation	49998.9 m	49998.9 m
Longitude Standard Deviation	49998.9 m	49998.9 m
IMU Temperature	39.5 °	38.3388 °
Depth	1.57395 m	0.847515 m
G Force	1.10925 g	0.818684 g

Untuk mengetahui keakuratan data, nilai pembacaan data dengan perlakuan. Hasil visualisasi dari perlakuan pada Sublocus dapat dilihat pada Gambar 4.14. Dari grafik ini melihatkan data fluktuasi akibat perlakuan untuk membuktikan data *orientation*.



s

Gambar 4. 14 Grafik Data Pengujian Menggunakan Dua Komputer (Raspberry Pi)

Pada pengujian ini menghasilkan *framerate* dengan rata-rata 23.675 fps. *Framerate* ini dapat dikatakan bagus dengan hasil mendekati 25 fps. Karena dengan *framerate* sebesar itu sudah dapat dilihat dengan baik.

Pada pengujian sistem keseluruhan terlihat perbedaan hasil framerate antara penggunaan *Personal Computer* dan Raspberry Pi.

Tabel 4. 6 Hasil *Framerate*

Metode	Rata-rata (fps)
Personal Computer	62.7228
Raspberry Pi	25.11092
Personal Computer (bersaamaan)	50.2999
Raspberry Pi (bersamaan)	23.67513

Rata-rata *Personal Computer* = 56.510fps
Rata-rata Raspberry Pi = 24.392 fps
Selisih rata-rata = 32.118 fps
Presentase perbandingan $framerate$=32.118/56510 ×100%
= 56.83%

Framerate yang dihasilkan pada penggunaan *Personal Computer* yakni mencapai 50.2999 *fps*. Sedangkan pada penggunaan Raspberry Pi *framerate* yang dihasilkan lebih kecil yakni 23.67513 *fps*. Sensor Sublocus mempunyai frekuensi kecepatan data sebesar 20Hz yang berarti sensor ini menghasilkan *framerate* 20 *fps* setiap Packet ID-nya. Pada perancangan perangkat lunak, ada 2 paket ID yang digunakan yaitu ANPP *Packet* 20 dan ANPP *Packet* 28. Dapat dikatakan kecepatan data mencapai 40 Hz atau 40 *fps* untuk 2 paket ID. Melihat *framerate* yang dihasilkan pada pengujian lewat *Personal Computer* yakni 50.2999 *fps* sedikit lebih besar dari *framerate* yang dimiliki Sublocus. Namun pada pengujian lewat Raspberry Pi, *framerate* yang dihasilkan setengah lebih kecil dari Sublocus. Hasil pengujian memiliki perbedaan *framerate* sebesar 56.83% antara *Personal Computer* dengan Raspberry Pi. Hal tersebut dikarenakan adanya perbedaan prosesor pada komputer. *Personal Computer* yang digunakan menggunakan prosesor intel core i3 yang mempunyai CPU *clock* sebesar 1.80 GHz sedangkan pada Raspberry Pi prosesor yang dimiliki mempunyai CPU *clock* sebesar 1.2

GHz. Semakin besar CPU *clock* semakin besar *framerate* yang dihasilkan pada tampilan GUI.

4.5. Validasi Pengujian

Setelah melakukan tiga metode pengujian penggunaan komputer. Pengujian meliputi pengujian GUI dengan menggunakan *Personal Computer* (PC), pengujian GUI dengan menggunakan *Single Board Computer* Raspberry Pi, dan pengujian GUI dengan menggunakan PC dan Raspberry Pi secara bersamaaan. Berikut adalah hasil validasi pengujian yang telah dilakukan:

Tabel 4. 7 Validasi Pengujian

Target	Hasil
Dapat membaca data dari sensor inersia Sublocus yakni pada ANPP *Packet* 20 dan *Packet* 28.	Pengujian GUI dapat membaca data dari sensor inersia Sublocus yakni pada ANPP *Packet* 20 dan *Packet* 28.
Dapat menampilkan data ANPP *Packet 20* dan *Packet* 28 sebanyak 19 data.	GUI yang telah dibuat dapat menampilkan data ANPP *Packet 20* dan *Packet* 28 sebanyak 19 data. Data yang ditampilan yakni *roll, pitch, yaw, latitude, longitude, velocity north, east, down, body acceleration X, Y, Z, angular velocity X, Y, Z, latitude* dan *longitude standard deviation,* IMU *temperature, depth,* dan *GForce.*
GUI yang dibuat dapat beroperasi di sistem operasi Windows pada *Personal Computer* dan Raspbian pada Raspberry Pi.	GUI yang telah dibuat dapat diuji pada sistem operasi Windows pada *Personal Computer* dan Raspbian pada Raspberry Pi.
GUI yang dibuat mampu menghasilkan *framerate* sebesar 25fps pada Raspberry Pi yang ditampilkan pada *underwater monitor* dan sebesar 50fps pada *Personal Computer.*	Pengujian menghasilkan *framerate* rata-rata sebesar 24.392 fps Raspberry Pi yang ditampilkan pada *underwater monitor* dan sebesar 56.510fps pada *Personal Computer.*

Tabel 4. 8 Validasi Pengujian (lanjutan)

Target	Hasil
GUI yang dibuat dapat ditampilkan pada layar yang memiliki resolusi sebesar 1366x768 pada *Personal Computer* dan 1280x768 pada *underwater monitor.*	GUI dapat diuji pada layar yang memiliki resolusi sebesar 1366x768 pada *Personal Computer* dan 1280x768 pada *underwater monitor.*

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

Pada Tugas Akhir ini telah didesain *Graphical User Interface* sistem antarmuka antara sensor inersia Sublocus dan Raspberry Pi. GUI tersebut dapat membaca dan menampilkan data dari sensor inersia Sublocus yang diperlukan kendaraan selam. GUI yang telah dibuat dapat dijalankan di Personal Computer dengan *framerate* rata-rata 56.510 *fps* dan Raspberry Pi 3 model B dengan *framerate* rata-rata 24.392 *fps*. *Framerate* GUI pada *Personal Computer* lebih besar 56.83 % dari *framerate* GUI pada Raspberry Pi 3 model B. Hal ini karena semakin besar CPU *clock* semakin besar *framerate* yang dihasilkan pada tampilan GUI.

Pengujian menggunakan PC *framerate* yang dihasilkan pada GUI sangat besar akibat pengaruh CPU *clock* dan resolusi layar. Pengujian ini memiliki keuntungan yakni GUI yangdiampilkan sangat bagus. Namun penggunaan PC memiliki kekurangan yakni tidak dapat mendukung sistem pada KTBA. Sedangkan pengujian menggunakan Raspberry Pi *framerate* yang dihasilkan lebih kecil. Namun dari kedua komputer ini, komputer yang paling sesuai digunakan untuk kendaraan selam ini adalah Raspberry Pi, karena komputer ini berukuran kecil dan fitur yang dimiliki sudah sangat mencukupi untuk keperluan pemrosesan data sistem kendaraan ini meskipun tampilan GUI pada Raspberry Pi tidak sebagus pada PC, karena *framerate* yang dihasilkan sudah dapat dilihat dengan baik oleh *user*.

Berdasarkan perencanaan dan pengujian yang dilakukan oleh penulis, maka pengembangan selanjutnya dari sistem ini adalah sistem antarmuka yang selain menampilkan data dalam bentuk angka akan lebih bagus ditampilkan dalam bentuk grafis. Karena pada perancangan yang sudah dilakukan hanya menampilkan angka dan memproses data. maka pengembangan selanjutnya diharapkan mampu melakukan antarmuka dalam bentuk GUI yang menampilkan grafis dan visualisasi sehingga dapat mempermudah pengguna dalam mengoperasikan kendaraan selam.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] -, "QCheckBox Class Reference", [Online]. Available: http://pyqt.sourceforge.net/Docs/PyQt4/qcheckbox.html [Accessed 26 Maret 2018].

[2] -, "QColor Class Reference", [Online]. Available: http://pyqt.sourceforge.net/Docs/PyQt4/qcolor.html [Accessed 26 Maret 2018].

[3] -, "QFile Class Reference", [Online]. Available: http://pyqt.sourceforge.net/Docs/PyQt4/qfile.html [Accessed 26 Maret 2018].

[4] -, "QIODevice Class Reference", [Online]. Available: http://pyqt.sourceforge.net/Docs/PyQt4/qiodevice.html [Accessed 26 Maret 2018].

[5] -, "QLabel Class Reference", [Online]. Available: http://pyqt.sourceforge.net/Docs/PyQt4/qlabel.html [Accessed 26 Maret 2018].

[6] -, "QLineEdit Class Reference", [Online]. Available: http://pyqt.sourceforge.net/Docs/PyQt4/qlineedit.html [Accessed 26 Maret 2018].

[7] -, "QPainter Class Reference", [Online]. Available: http://pyqt.sourceforge.net/Docs/PyQt4/qpainter.html [Accessed 26 Maret 2018].

[8] -, "QTimer Class Reference", [Online]. Available: http://pyqt.sourceforge.net/Docs/PyQt4/qtimer.html [Accessed 26 Maret 2018].

[9] -, "Raspbian", [Online]. Available: https://www.raspberrypi.org/documentation/raspbian/ [Accessed 25 Maret 2018].

[10] -, "Shallow Water Combat Submersible (SWCS/SDV)" Engtek SubSea Systems, 2017.

[11] -,"Poseidon Reference Manual," 2016. [Online]. Available: http://www.advancednavigation.com.au/sites/advancednavigation.com.au/files/poseidon_reference_manual_0.pdf [Accessed 25 November 2017].

[12] -,"RS422," [Online]. Available: http://www.futureelectronics.com/en/Signal-Interface/rs-422.aspx [Accessed 23 Maret 2018].

[13] -,"Sublocus Reference Manual," 2015. [Online]. Available: http://www.advancednavigation.com.au/sites/advancednavigation.com.au/files/sublocus_reference_manual.pdf [Accessed 20 November 2017].
[14] -,"Sublocus," 2015. [Online]. Available: http://www.advancednavigation.com.au/sites/advancednavigation.com.au/files/SublocusDatasheet.pdf. [Accessed 20 November 2017].
[15] -,"U-RS422," 2016. [Online]. Available: http://www.chronos.com.tw/files/spec/u-rs422.pdf [Accessed 23 Januari 2018].
[16] -,"U-RS422TB," 2016. [Online]. Available: http://www.chronos.com.tw/files/spec/u-rs422tb.pdf [Accessed 23 Januari 2018].
[17] Cahyo, Budi., Rochim, Adian Fatchur., Widianto, Eko Didik., "Rancang Bangun Purwarupa Sistem Navigasi Tanpa Awak untuk Kapal," Jurnal Teknologi dan Sistem Komputer, Vol.4, No.1, 2016.
[18] Christian, Frendy, "Modul Pembelajaran Raspberry," Universitas Sanata Dharma, 2017.
[19] Dive and See, "DNC-10V," [Online]. Available: http://www.diveandsee.com/products/underwater-monitors/dnc-10v [Accessed 25 Maret 2018].
[20] Grewal, M. S., Weill, L. R., & Andrews, A. P, "Global Positioning Systems, Inertial Navigation, and Integration," John Wiley & Sons (Vol. 2), 2000.
[21] Harry, A.Jackson, "Fundamental of Submarine Concept Design," Massachusetts Institute of Technology, SNAME Transactions, Vol. 100, pp. 419-448, 1992.
[22] Ishibashi, S., Tsukioka, S., Sawa, T., Yoshida, H., Hyakudome, T., Tahara, J., Ishikawa, A., "The Rotation Control System To Improve The Accuracy of An Inertial Navigation System Installed in An Autonomous Underwater Vehicle. International Symposium on Underwater Technology," UT- International Workshop on Scientific Use of Submarine Cables and Related Technologies, 2007.
[23] Kennedy, M., "The Global Positioning System and Its Applications," Indian Institute of Techology, 2010.
[24] Lykov, A., Tarpley, W., & Volkov, A, "GPS + Inertial Sensor Fusion Group Members," Bradley University ECE Department, 2014.

[25] Mansour, Alaa., Liu, Donald., “Strength of Ships and Ocean Structure,” The Society of Naval Architects and Marine Engineers, 2008.
[26] Mark Summerfield, “Rapid GUI Proramming with Python and Qt,” Parentice Hill, 2007.
[27] Nugroho, Wibowo, “Perancangan Kapal Selam Berdasarkan Kajian Berat, Daya Apung & Stabilitas Statisnya,” Marine Structural Monitoring/Hydroelasticity Group Indonesian Hydrodynamics Laboratory UPT- BPPH, BPPT, Surabaya, 2007.
[28] Qt, F. Pemrograman Aplikasi GUI, (April), 1–20, 2013.
[29] Rezaifard, E., & Abbasi, P, “Inertial Navigation System Calibration Using GPS Based on Extended Kalman Filter,” ICEE, 2017.
[30] Suyadi, “Komunikasi Serial dan Port Serial (COM)”, Teknik Informatika UMS, 2012.
[31] Xu, J., He, H., Qin, F., & Chang, L, “A Novel Autonomous Initial Alignment Method for Strapdown Inertial Navigation System,” IEEE, 2017.
[32] Yudhasasmita, Pekik, “Kendaraan Tempur Taktis Bawah Air (Studi Kasus Elite Kopaska TNI AL),” Inosains Vol.7, No.2, 2012.

Halaman ini sengaja dikosongkan

LAMPIRAN

A-1 Program Serial Reading

```
import serial
from PyQt4 import QtCore
from PyQt4.QtCore import pyqtSlot
from ANPP import ANPacketDecoder, ANPacket20, ANPacket28
import math

class SerialReading(QtCore.QObject):
    ReadPacket_Orientation = QtCore.pyqtSignal(list)
    ReadPacket_Position = QtCore.pyqtSignal(list)
    ReadPacket_NEDVelocity = QtCore.pyqtSignal(list)
    ReadPacket_BodyAcceleration = QtCore.pyqtSignal(list)
    ReadPacket_AngularVelocity = QtCore.pyqtSignal(list)
    ReadPacket_PositionStandardDeviation =
QtCore.pyqtSignal(list)
    Readpacket_IMUtemp = QtCore.pyqtSignal(float)
    ReadPacket_SystemFailure = QtCore.pyqtSignal(bool)
    ReadPacket_PreassureOverange = QtCore.pyqtSignal(bool)
    ReadPacket_MinimumTemperature = QtCore.pyqtSignal(bool)
    ReadPacket_MaximumTemperature = QtCore.pyqtSignal(bool)
    ReadPacket_HeadingInitialised = QtCore.pyqtSignal(bool)
    ReadPacket_ExternalVelocity = QtCore.pyqtSignal(bool)
    ReadPacket_GPSFailure = QtCore.pyqtSignal(bool)
    ReadPacket_GPSFixStatus = QtCore.pyqtSignal(bool)
    ReadPacket_GPSFixType = QtCore.pyqtSignal(int)

    def __init__(self, portName, baudrate, timeout,
parent=None):
        super(SerialReading, self).__init__(parent)
        self.portName = portName
        self.baudrate = baudrate
        self.timeout = timeout

        self.ANPD = ANPacketDecoder()
        self.port = None
        self.isReadingStopped = False

        try:
            self.port = serial.Serial(self.portName,

baudrate=self.baudrate,

parity=serial.PARITY_NONE,

stopbits=serial.STOPBITS_ONE,
```

```
bytesize=serial.EIGHTBITS,
                                        writeTimeout=0,
                                        timeout=self.timeout,
                                        rtscts=False,
                                        dsrdtr=False,
                                        xonxoff=False)
        except:
            print ("Can't open Serial Port")



    def write(self, data):
        if self.port.isOpen():
            try:
                self.port.write(data)
                self.port.flush()
            except serial.SerialException:
                print "Error on Writing"
        else:
            print "Port is Closed"

    def read(self):
        if self.port is None:
            return

        if not self.port.isOpen():
            try:
                self.port = serial.Serial(self.portName,

baudrate=self.baudrate)
            except serial.SerialException:
                print "Error on Opening"
                return

        self.readSublocus()

    def readSublocus(self):
        bytes_waiting = self.port.inWaiting()
        if bytes_waiting > 0:
            serialBuffer = self.port.read(bytes_waiting)
            serialBuffer_int = [ord(x)for x in serialBuffer]
            for i in range(len(serialBuffer_int)):
                if self.ANPD.bufferLength <
len(self.ANPD.buffer):
                    self.ANPD.buffer[self.ANPD.bufferLength]
= serialBuffer_int[i]
                    self.ANPD.bufferLength += 1

            packet = self.ANPD.packetDecode()
```

```
            while packet is not None:
                if packet.id == 20:
                    if packet.length == 112:
                        anPacket20 = ANPacket20(packet.data)

                        roll = anPacket20.orientation[0]
                        pitch = anPacket20.orientation[1]
                        yaw = anPacket20.orientation[2]

                        latitude = anPacket20.position[0]
                        longitude = anPacket20.position[1]

                        vNorth = anPacket20.velocity[0]
                        vEast = anPacket20.velocity[1]
                        vDown = anPacket20.velocity[2]

                        aX = anPacket20.acceleration[0]
                        aY = anPacket20.acceleration[1]
                        aZ = anPacket20.acceleration[2]
                        gForce = anPacket20.gForce

                        AVX = anPacket20.angularVelocity[0]
                        AVY = anPacket20.angularVelocity[1]
                        AVZ = anPacket20.angularVelocity[2]

                        latSD =
anPacket20.positionStandardDeviation[0]
                        longSD =
anPacket20.positionStandardDeviation[1]
                        depth = anPacket20.depth

                        system failure =
anPacket20.systemFailure
                        preassure_overrange =
anPacket20.pressureOverRange
                        mintemp =
anPacket20.minimumTemperatureAlarm
                        maxtemp =
anPacket20.maximumTemperatureAlarm
                        heading =
anPacket20.headingInitialised
                        exvelo =
anPacket20.externalVelocityActive
                        gpsfail = anPacket20.gnssFailure
                        gpsfix = anPacket20.gnssFix

                        gpsfixstatus =
anPacket20.gnssFixType
```

```
self.ReadPacket_Orientation.emit([roll, pitch, yaw])

self.ReadPacket_Position.emit([latitude, longitude])

self.ReadPacket_NEDVelocity.emit([vNorth, vEast, vDown])

self.ReadPacket_BodyAcceleration.emit([aX, aY, aZ, gForce])

self.ReadPacket_AngularVelocity.emit([AVX, AVY, AVZ])

self.ReadPacket_PositionStandardDeviation.emit([latSD,longSD
, depth])

self.ReadPacket_SystemFailure.emit(system_failure)

self.ReadPacket_PreassureOverange.emit(preassure overrange)

self.ReadPacket_MinimumTemperature.emit(mintemp)

self.ReadPacket_MaximumTemperature.emit(maxtemp)

self.ReadPacket_HeadingInitialised.emit(heading)

self.ReadPacket_ExternalVelocity.emit(exvelo)

self.ReadPacket_GPSFailure.emit(gpsfail)

self.ReadPacket_GPSFixStatus.emit(gpsfix)

self.ReadPacket_GPSFixType.emit(gpsfixstatus)

                elif packet.id == 28:
                    if packet.length == 48:
                        anPacket28 = ANPacket28(packet.data)

                        imu = anPacket28.imuTemperature

                        self.Readpacket_IMUtemp.emit(imu)


                packet = self.ANPD.packetDecode()
        else:
            pass  # Reserved for future use.

    @pyqtSlot()
    def startReading(self):
        while not self.isReadingStopped:
```

```
            QtCore.QCoreApplication.processEvents()
            self.read()

    @pyqtSlot()
    def stopReading(self):
        self.isReadingStopped = True
```

A-2 Program GUI dan *Data Processing*

```
import sys
import os
import math
import time
from PyQt4 import QtGui, QtCore
from PyQt4.QtCore import pyqtSlot, pyqtSignal
from SerialReading import SerialReading
from ANPP import ANPacket20


class Window(QtGui.QMainWindow):
    StartSublocusReading = pyqtSignal()
    StopSublocusReading = pyqtSignal()

    def __init__(self):
        super(Window, self).__init__()

        self.counter = 0
        self.totalpaint = 0.0

        self.logTimer = QtCore.QTimer(self)
        self.logTimer.setInterval(200)
        self.logTimer.timeout.connect(self.LogDatatoFile)
        self.logTimer.start()

        self.fps = 0
        self.fpsTimer = QtCore.QTimer(self)
        self.fpsTimer.setInterval(1000)
        self.fpsTimer.timeout.connect(self.CalculateFps)
        self.fpsTimer.start()

        self.ANPP20file = QtCore.QFile(self)
        self.ANPP20file.setFileName("D:/SEMESTER
5/TA/DOKUMENTASI/14-3-2018/Data5.csv")
        self.ANPP20file.open(QtCore.QIODevice.WriteOnly |
QtCore.QIODevice.Append)
        textstream = QtCore.QTextStream(self.ANPP20file)

textstream.__lshift__("fps,roll,pitch,yaw,latitude,longitude
,velocitynorth,velocityeast,velocitydown,bodyaccelerationX,b
odyaccelerationY,bodyaccelerationZ,angularvelocityX,angularv
elocityY,angularvelocityZ,latitudestandev,longitudestandev,i
mutemp,depth,gforce\n")
        self.roll = 0.0
        self.pitch = 0.0
        self.yaw = 0.0
        self.latitude = 0.0
        self.longitude = 0.0
```

```
        self.velocitynorth = 0.0
        self.velocittyeast = 0.0
        self.velocitydown = 0.0
        self.bodyaccelerationX = 0.0
        self.bodyaccelerationY = 0.0
        self.bodyaccelerationZ = 0.0
        self.angularvelocityX = 0.0
        self.angularvelocityY = 0.0
        self.angularvelocityZ = 0.0
        self.latitudestandev = 0.0
        self.longitudestandev = 0.0
        self.imutemp = 0.0
        self.depth = 0.0
        self.gforce = 0.0


        self.sublocusThread = QtCore.QThread()
        self.sublocusWorker = SerialReading("COM7", 115200,
10)

self.sublocusWorker.ReadPacket Orientation.connect(self.upda
teRollPitchYaw)

self.sublocusWorker.ReadPacket Position.connect(self.updateL
ongLat)

self.sublocusWorker.ReadPacket_NEDVelocity.connect(self.upda
teVeloNorthEastDown)

self.sublocusWorker.ReadPacket BodyAcceleration.connect(self
.updateBodyAcceleration)

self.sublocusWorker.ReadPacket_AngularVelocity.connect(self.
updateAngularVelocity)

self.sublocusWorker.ReadPacket PositionStandardDeviation.con
nect(self.updatePositionStandardDeviation)

self.sublocusWorker.Readpacket_IMUtemp.connect(self.updateIM
U)

self.sublocusWorker.ReadPacket SystemFailure.connect(self.sy
stem failure)

self.sublocusWorker.ReadPacket_PreassureOverange.connect(sel
f.preassure_overrange)

self.sublocusWorker.ReadPacket MinimumTemperature.connect(se
lf.min_temperature)
```

```
self.sublocusWorker.ReadPacket_MaximumTemperature.connect(se
lf.max temperature)

self.sublocusWorker.ReadPacket_HeadingInitialised.connect(se
lf.heading)

self.sublocusWorker.ReadPacket_ExternalVelocity.connect(self
.exteral velo)

self.sublocusWorker.ReadPacket_GPSFailure.connect(self.gps_f
ailure)

self.sublocusWorker.ReadPacket GPSFixStatus.connect(self.gps
 fix)

self.sublocusWorker.ReadPacket_GPSFixType.connect(self.updat
eGPSFixStatus)

self.StartSublocusReading.connect(self.sublocusWorker.startR
eading)

self.StopSublocusReading.connect(self.sublocusWorker.stopRea
ding)

self.sublocusWorker.moveToThread(self.sublocusThread)
        self.sublocusThread.start()

        self.text = "USER INTERFACE SENSOR INERSIA SUBLOCUS"
        self.status = "Status"
        self.rawsensor = "Raw Sensor"
        self.capfps = "fps: "
        self.imu = "IMU Temperature: "
        self.odepth = "Depth: "
        self.ogforce = "G Force: "
        self.orientation = "Orientation"
        self.oroll = "Roll: "
        self.opitch = "Pitch: "
        self.oyaw = "Yaw: "
        self.velocity = "Velocity"
        self.velonorth = "Velocity North: "
        self.veloeast = "Velocity East: "
        self.velodown = "Velocity Down: "
        self.position = "Position"
        self.plongitude = "Longitude: "
        self.platitude = "Latitude: "
        self.bodyacc = "Body Acceleration"
        self.bodyaccx = "Body Acceleration X: "
        self.bodyaccy = "Body Acceleration Y: "
```

```
        self.bodyaccz = "Body Acceleration Z: "
        self.av = "Angular Velocity"
        self.avx = "Angular Velocity X: "
        self.avy = "Angular Velocity Y: "
        self.avz = "Angular Velocity Z: "
        self.psd = "Position Standard Deviation"
        self.longsd = "Longitude Standard Deviation: "
        self.latsd = "Latitude Standard Deviation: "


        self.labelTitle = QtGui.QLabel(self)

        self.labelStatus = QtGui.QLabel(self)
        self.checkBox1 = QtGui.QCheckBox(self)
        self.checkBox2 = QtGui.QCheckBox(self)
        self.checkBox3 = QtGui.QCheckBox(self)
        self.checkBox4 = QtGui.QCheckBox(self)
        self.checkBox5 = QtGui.QCheckBox(self)
        self.checkBox6 = QtGui.QCheckBox(self)
        self.checkBox7 = QtGui.QCheckBox(self)
        self.checkBox8 = QtGui.QCheckBox(self)

        self.pic = QtGui.QLabel(self)
        self.pic.setGeometry(0, 30, 1000, 1000)
        self.pic.setPixmap(QtGui.QPixmap(os.getcwd() +
"/sublocus.png"))

        self.txtfps = QtGui.QLineEdit(self)
        self.txtGPSStatus = QtGui.QLineEdit(self)
        self.txtImu = QtGui.QLineEdit(self)
        self.txtGForce = QtGui.QLineEdit(self)
        self.txtDepth = QtGui.QLineEdit(self)
        self.txtRoll = QtGui.QLineEdit(self)
        self.txtPitch = QtGui.QLineEdit(self)
        self.txtYaw = QtGui.QLineEdit(self)
        self.txtLong = QtGui.QLineEdit(self)
        self.txtLat = QtGui.QLineEdit(self)
        self.txtVeloNorth = QtGui.QLineEdit(self)
        self.txtVeloEast = QtGui.QLineEdit(self)
        self.txtVeloDown = QtGui.QLineEdit(self)
        self.txtBodyAccX = QtGui.QLineEdit(self)
        self.txtBodyAccY = QtGui.QLineEdit(self)
        self.txtBodyAccZ = QtGui.QLineEdit(self)
        self.txtAngularVeloX = QtGui.QLineEdit(self)
        self.txtAngularVeloY = QtGui.QLineEdit(self)
        self.txtAngularVeloZ = QtGui.QLineEdit(self)
        self.txtLongSD = QtGui.QLineEdit(self)
        self.txtLatSD = QtGui.QLineEdit(self)
```

```
        self.setGeometry(0, 0, 500, 300)
        self.setWindowTitle("User Interface INS")
        self.home()

    def home(self):
        self.txtGPSStatus.move(300, 270)
        self.txtGPSStatus.resize(200, 20)
        self.txtGPSStatus.setText("no status")

        self.txtfps.move(70, 330)
        self.txtfps.resize(80, 30)
        self.txtfps.setText("0")

        self.txtImu.move(400, 430)
        self.txtImu.resize(100, 30)
        self.txtImu.setText("0")

        self.txtDepth.move(400, 560)
        self.txtDepth.resize(100, 30)
        self.txtDepth.setText("0")

        self.txtGForce.move(400, 660)
        self.txtGForce.resize(100, 30)
        self.txtGForce.setText("0")

        self.txtRoll.move(800, 100)
        self.txtRoll.resize(100, 30)
        self.txtRoll.setText("0")

        self.txtPitch.move(800, 150)
        self.txtPitch.resize(100, 30)
        self.txtPitch.setText("0")

        self.txtYaw.move(800, 200)
        self.txtYaw.resize(100, 30)
        self.txtYaw.setText("0")

        self.txtVeloNorth.move(800, 350)
        self.txtVeloNorth.resize(100, 30)
        self.txtVeloNorth.setText("0")

        self.txtVeloEast.move(800, 400)
        self.txtVeloEast.resize(100, 30)
        self.txtVeloEast.setText("0")

        self.txtVeloDown.move(800, 450)
        self.txtVeloDown.resize(100, 30)
        self.txtVeloDown.setText("0")
```

```
        self.txtLong.move(800, 580)
        self.txtLong.resize(100, 30)
        self.txtLong.setText("0")

        self.txtLat.move(800, 630)
        self.txtLat.resize(100, 30)
        self.txtLat.setText("0")

        self.txtBodyAccX.move(1240, 100)
        self.txtBodyAccX.resize(100, 30)
        self.txtBodyAccX.setText("0")

        self.txtBodyAccY.move(1240, 150)
        self.txtBodyAccY.resize(100, 30)
        self.txtBodyAccY.setText("0")

        self.txtBodyAccZ.move(1240, 200)
        self.txtBodyAccZ.resize(100, 30)
        self.txtBodyAccZ.setText("0")

        self.txtAngularVeloX.move(1190, 350)
        self.txtAngularVeloX.resize(150, 30)
        self.txtAngularVeloX.setText("0")

        self.txtAngularVeloY.move(1190, 400)
        self.txtAngularVeloY.resize(150, 30)
        self.txtAngularVeloY.setText("0")

        self.txtAngularVeloZ.move(1190, 450)
        self.txtAngularVeloZ.resize(150, 30)
        self.txtAngularVeloZ.setText("0")

        self.txtLongSD.move(1240, 580)
        self.txtLongSD.resize(100, 30)
        self.txtLongSD.setText("0")

        self.txtLatSD.move(1240, 630)
        self.txtLatSD.resize(100, 30)
        self.txtLatSD.setText("0")

        self.showMaximized()

        self.checkBox1.setText("System Failure")
        self.checkBox1.setGeometry(10, 100, 200, 20)

self.checkBox1.stateChanged.connect(self.system_failure)
        self.checkBox2.setText("Preassure Overrange")
        self.checkBox2.setGeometry(10, 150, 200, 20)
```

```
self.checkBox2.stateChanged.connect(self.preassure_overrange
)
        self.checkBox3.setText("Minimum Temperature Alarm")
        self.checkBox3.setGeometry(10, 200, 200, 20)

self.checkBox3.stateChanged.connect(self.min_temperature)
        self.checkBox4.setText("Maximum Temperature Alarm")
        self.checkBox4.setGeometry(10, 250, 200, 20)

self.checkBox4.stateChanged.connect(self.max_temperature)

        self.checkBox5.setText("Heading Initialised")
        self.checkBox5.setGeometry(300, 100, 200, 20)
        self.checkBox5.stateChanged.connect(self.heading)
        self.checkBox6.setText("External Velocity Active")
        self.checkBox6.setGeometry(300, 150, 200, 20)

self.checkBox6.stateChanged.connect(self.exteral velo)
        self.checkBox7.setText("GPS Failure")
        self.checkBox7.setGeometry(300, 200, 200, 20)

self.checkBox7.stateChanged.connect(self.gps_failure)
        self.checkBox8.setText("GPS Fix Status")
        self.checkBox8.setGeometry(300, 250, 200, 20)
        self.checkBox8.stateChanged.connect(self.gps fix)

        self.show()


    @pyqtSlot(bool)
    def system_failure(self, packet):
        self.checkBox1.setChecked(packet)

    @pyqtSlot(bool)
    def preassure_overrange(self, packet):
        self.checkBox2.setChecked(packet)

    @pyqtSlot(bool)
    def min_temperature(self, packet):
        self.checkBox3.setChecked(packet)

    @pyqtSlot(bool)
    def max_temperature(self, packet):
        self.checkBox4.setChecked(packet)

    @pyqtSlot(bool)
    def heading(self, packet):
        self.checkBox5.setChecked(packet)
```

```
    @pyqtSlot(bool)
    def exteral_velo(self, packet):
        self.checkBox6.setChecked(packet)

    @pyqtSlot(bool)
    def gps_failure(self, packet):
        self.checkBox7.setChecked(packet)

    @pyqtSlot(bool)
    def gps_fix(self, packet):
        self.checkBox8.setChecked(packet)

    @pyqtSlot(int)
    def updateGPSFixStatus(self, packet):
        if packet == ANPacket20.GNSS_NO_FIX :
            self.txtGPSStatus.setText("No Fix")
        elif packet == ANPacket20.GNSS_2D_FIX :
            self.txtGPSStatus.setText("2D Fix")
        elif packet == ANPacket20.GNSS_3D_FIX :
            self.txtGPSStatus.setText("3D Fix")
        elif packet == ANPacket20.GNSS_SBAS_FIX :
            self.txtGPSStatus.setText("SBAS Fix")
        elif packet == ANPacket20.GNSS_OMNISTAR_FIX :
            self.txtGPSStatus.setText("Omnistar Fix")
        elif packet == ANPacket20.GNSS_DIFFERENTIAL_FIX :
            self.txtGPSStatus.setText("Differential Fix")
        elif packet == ANPacket20.GNSS_RTK_FLOAT_FIX :
            self.txtGPSStatus.setText("RTK Float Fix")
        elif packet == ANPacket20.GNSS_RTK_FIXED_FIX :
            self.txtGPSStatus.setText("RTK Fixed Fix")
        else :
            self.txtGPSStatus.setText("no data")

    def keyPressEvent(self, event):
        if event.key() == QtCore.Qt.Key_1:
            self.updateRollPitchYaw([10, 20, 30])

        if event.key() == QtCore.Qt.Key_2:
            self.updateLongLat([10, 20])

        if event.key() == QtCore.Qt.Key_3:
            self.updateVeloNorthEastDown([10, 20])

        if event.key() == QtCore.Qt.Key_4:
            self.updateBodyAcceleration([10, 20])

    def closeEvent(self, QCloseEvent):
        if self.sublocusThread.isRunning():
            self.StopSublocusReading.emit()
```

```
            self.sublocusThread.exit()
            self.sublocusThread.wait()

    def paintEvent(self, event):
        self.counter += 1
        startTime = time.time()

        painter = QtGui.QPainter(self)

painter.setBackground(QtGui.QBrush(QtCore.Qt.Dense1Pattern))
        painter.eraseRect(self.rect())

        qp = QtGui.QPainter()
        qp.begin(self)
        self.drawLines(qp)
        qp.end()

        qp = QtGui.QPainter()
        qp.begin(self)
        self.drawRectangles(qp)
        qp.end()

        qp = QtGui.QPainter()
        qp.begin(self)
        self.drawText(event, qp)
        qp.end()

        paintduration = time.time()-startTime
        self.totalpaint+= paintduration


    def drawLines(self, qp):

        pen = QtGui.QPen(QtCore.Qt.white, 2,
QtCore.Qt.DashLine)

        qp.setPen(pen)
        qp.drawLine(550, 270, 1360, 270)

        pen.setStyle(QtCore.Qt.DashLine)
        qp.setPen(pen)
        qp.drawLine(550, 500, 1360, 500)

        pen.setStyle(QtCore.Qt.DashLine)
        qp.setPen(pen)
        qp.drawLine(0, 320, 550, 320)

        pen.setStyle(QtCore.Qt.DashLine)
        qp.setPen(pen)
```

```
        qp.drawLine(0, 40, 2000, 40)

        pen.setStyle(QtCore.Qt.DashLine)
        qp.setPen(pen)
        qp.drawLine(550, 40, 0, 100000)

        pen.setStyle(QtCore.Qt.DashLine)
        qp.setPen(pen)
        qp.drawLine(950, 40, 0, 100000)


    def drawRectangles(self, qp):

        color = QtGui.QColor(0, 0, 0)
        color.setNamedColor('#d4d4d4')
        qp.setPen(color)

        #Draw Top
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(350, 0, 720, 40)

        #Draw Status
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(210, 50, 140, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.lightGray))
        qp.drawRect(5, 90, 260, 210)
        qp.setBrush(QtGui.QColor(QtCore.Qt.lightGray))
        qp.drawRect(280, 90, 250, 210)

        #FPS
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(10, 330, 50, 30)

        #IMU
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(400, 340, 130, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(400, 390, 130, 30)

        #Depth
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(400, 520, 80, 30)

        #G Force
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(400, 620, 80, 30)

        # Orientation
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
```

```
        qp.drawRect(680, 50, 150, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(590, 100, 180, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(590, 150, 180, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(590, 200, 180, 30)

        # Body Acceleration
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(1060, 50, 225, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(990, 100, 180, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(990, 150, 180, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(990, 200, 180, 30)

        # Velocity
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(690, 300, 125, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(590, 350, 180, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(590, 400, 180, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(590, 450, 180, 30)

        # Angular Velocity
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(1060, 300, 210, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(990, 350, 180, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(990, 400, 180, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(990, 450, 180, 30)

        #Position
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(700, 530, 110, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(590, 580, 180, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(590, 630, 180, 30)

        #Position Standard Deviation
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(1010, 530, 310, 30)
```

```
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(990, 580, 220, 30)
        qp.setBrush(QtGui.QColor(QtCore.Qt.black))
        qp.drawRect(990, 630, 220, 30)

    def drawText(self, event, qp):
        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Safety', 20))
        qp.drawText(QtCore.QPointF(375, 30), self.text)

        qp.setPen(QtGui.QColor(QtCore.Qt.lightGray))
        qp.setFont(QtGui.QFont('Century Gothic', 15))
        qp.drawText(QtCore.QPointF(250, 70), self.status)

        qp.setPen(QtGui.QColor(QtCore.Qt.yellow))
        qp.setFont(QtGui.QFont('Century Gothic', 15))
        qp.drawText(QtCore.QPointF(410, 360),
self.rawsensor)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(20, 350), self.capfps)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(410, 410), self.imu)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(410, 540), self.odepth)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(410, 640), self.ogforce)

        qp.setPen(QtGui.QColor(QtCore.Qt.yellow))
        qp.setFont(QtGui.QFont('Century Gothic', 15))
        qp.drawText(QtCore.QPointF(700, 70),
self.orientation)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(600, 120), self.oroll)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(600, 170), self.opitch)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
```

```
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(600, 220), self.oyaw)

        qp.setPen(QtGui.QColor(QtCore.Qt.yellow))
        qp.setFont(QtGui.QFont('Century Gothic', 15))
        qp.drawText(QtCore.QPointF(710, 320), self.velocity)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(600, 370),
self.velonorth)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(600, 420), self.veloeast)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(600, 470), self.velodown)

        qp.setPen(QtGui.QColor(QtCore.Qt.yellow))
        qp.setFont(QtGui.QFont('Century Gothic', 15))
        qp.drawText(QtCore.QPointF(720, 550), self.position)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(600, 600),
self.plongitude)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(600, 650),
self.platitude)

        qp.setPen(QtGui.QColor(QtCore.Qt.yellow))
        qp.setFont(QtGui.QFont('Century Gothic', 15))
        qp.drawText(QtCore.QPointF(1080, 70), self.bodyacc)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(1000, 120),
self.bodyaccx)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(1000, 170),
self.bodyaccy)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
```

```
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(1000, 220),
self.bodyaccz)

        qp.setPen(QtGui.QColor(QtCore.Qt.yellow))
        qp.setFont(QtGui.QFont('Century Gothic', 15))
        qp.drawText(QtCore.QPointF(1080, 320), self.av)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(1000, 370), self.avx)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(1000, 420), self.avy)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(1000, 470), self.avz)

        qp.setPen(QtGui.QColor(QtCore.Qt.yellow))
        qp.setFont(QtGui.QFont('Century Gothic', 15))
        qp.drawText(QtCore.QPointF(1030, 550), self.psd)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(1000, 600), self.longsd)

        qp.setPen(QtGui.QColor(QtCore.Qt.white))
        qp.setFont(QtGui.QFont('Century Gothic', 10))
        qp.drawText(QtCore.QPointF(1000, 650), self.latsd)

    @pyqtSlot()
    def LogDatatoFile(self):
        if self.ANPP20file.isOpen():
            textstream = QtCore.QTextStream(self.ANPP20file)
            textstream.  lshift  (self.fps)
            textstream.__lshift__(",")
            textstream.__lshift__(self.roll)
            textstream.__lshift__(",")
            textstream.  lshift  (self.pitch)
            textstream.__lshift__(",")
            textstream.  lshift  (self.yaw)
            textstream.__lshift__(",")
            textstream.__lshift__(self.latitude)
            textstream.__lshift__(",")
            textstream.  lshift  (self.longitude)
            textstream.__lshift__(",")
            textstream.__lshift__(self.velocitynorth)
```

```
            textstream.__lshift__(",")
            textstream.__lshift__(self.velocittyeast)
            textstream.__lshift__(",")
            textstream.__lshift__(self.velocitydown)
            textstream.__lshift__(",")
            textstream.__lshift__(self.bodyaccelerationX)
            textstream.__lshift__(",")
            textstream.__lshift__(self.bodyaccelerationY)
            textstream.__lshift__(",")
            textstream.__lshift__(self.bodyaccelerationZ)
            textstream.__lshift__(",")
            textstream.__lshift__(self.angularvelocityX)
            textstream.__lshift__(",")
            textstream.__lshift__(self.angularvelocityY)
            textstream.__lshift__(",")
            textstream.__lshift__(self.angularvelocityZ)
            textstream.__lshift__(",")
            textstream.__lshift__(self.latitudestandev)
            textstream.__lshift__(",")
            textstream.__lshift__(self.longitudestandev)
            textstream.__lshift__(",")
            textstream.__lshift__(self.imutemp)
            textstream.__lshift__(",")
            textstream.__lshift__(self.depth)
            textstream.__lshift__(",")
            textstream.__lshift__(self.gforce)
            textstream.__lshift__("\n")

    @pyqtSlot()
    def CalculateFps(self):
        if self.totalpaint > 0:
            self.fps = float(self.counter) / self.totalpaint
            self.txtfps.setText(str(round(self.fps,2)))
        self.counter = 0
        self.totalpaint = 0

    @pyqtSlot(list)
    def updateRollPitchYaw(self, packet):
        roll_rad = packet[0]
        pitch_rad = packet[1]
        yaw rad = packet[2]

        roll = roll rad * 180 / math.pi
        pitch = pitch_rad * 180 / math.pi
        yaw = yaw_rad * 180 / math.pi

        self.roll = roll
        self.pitch = pitch
        self.yaw = yaw
```

```
        self.txtRoll.setText(str(round(roll,3)))
        self.txtPitch.setText(str(round(pitch,3)))
        self.txtYaw.setText(str(round(yaw,3)))


    @pyqtSlot(list)
    def updateLongLat(self, packet):
        latitude_rad = packet[0]
        longitude_rad = packet[1]

        latitude = latitude_rad * 180 / math.pi
        longitude = longitude_rad * 180 / math.pi

        self.latitude = latitude
        self.longitude  = longitude

        self.txtLong.setText(str(longitude))
        self.txtLat.setText(str(latitude))


    @pyqtSlot(list)
    def updateVeloNorthEastDown(self, packet):
        velonorth_ms = packet[0]
        veloeast_ms = packet[1]
        velodown_ms = packet[2]

        velonorth = velonorth_ms * 1.9
        veloeast = veloeast_ms * 1.9
        velodown = velodown_ms * 1.9

        self.velocitynorth = velonorth
        self.velocittyeast = veloeast
        self.velocitydown = velodown

        self.txtVeloNorth.setText(str(round(velonorth,3)))
        self.txtVeloEast.setText(str(round(veloeast,3)))
        self.txtVeloDown.setText(str(round(velodown,3)))


    @pyqtSlot(list)
    def updateBodyAcceleration(self, packet):
        bodyaccX = packet[0]
        bodyaccY = packet[1]
        bodyaccZ = packet[2]
        gForce = packet[3]

        self.bodyaccelerationX = bodyaccX
        self.bodyaccelerationY = bodyaccY
```

```
        self.bodyaccelerationZ  = bodyaccZ
        self.gforce = gForce

        self.txtBodyAccX.setText(str(round(bodyaccX,3)))
        self.txtBodyAccY.setText(str(round(bodyaccY,3)))
        self.txtBodyAccZ.setText(str(round(bodyaccZ,3)))
        self.txtGForce.setText(str(round(gForce,5)))


    @pyqtSlot(list)
    def updateAngularVelocity(self, packet):
        AVX = packet[0]
        AVY = packet[1]
        AVZ = packet[2]

        self.angularvelocityX = AVX
        self.angularvelocityY = AVY
        self.angularvelocityZ = AVZ

        self.txtAngularVeloX.setText(str(round(AVX,15)))
        self.txtAngularVeloY.setText(str(round(AVY,15)))
        self.txtAngularVeloZ.setText(str(round(AVZ,15)))

    @pyqtSlot(list)
    def updatePositionStandardDeviation(self, packet):
        latSD = packet[0]
        longSD = packet[1]
        depth = packet[2]

        self.latitudestandev = latSD
        self.longitudestandev = longSD
        self.depth = depth

        self.txtLatSD.setText(str(latSD))
        self.txtLongSD.setText(str(longSD))
        self.txtDepth.setText(str(depth))

    @pyqtSlot(float)
    def updateIMU(self, packet):
        imu = packet

        self.imutemp = imu

        self.txtImu.setText(str(round(imu,2)))


def run():
    app = QtGui.QApplication(sys.argv)
    mainWindow = Window()
```

```
    mainWindow.StartSublocusReading.emit()

    sys.exit(app.exec ())


run()
```

A-3 Program ANPP

```
import struct
import ctypes


def bytes_to_float(listBytes, startIndex):
    four_bytes = listBytes[startIndex:startIndex + 4]
    float_num = struct.pack('4B', *four_bytes)
    float_num = struct.unpack('f', float_num)
    return float_num[0]


def bytes_to_double(listBytes, startIndex):
    eight_bytes = listBytes[startIndex:startIndex + 8]
    double_num = struct.pack('8B', *eight_bytes)
    double_num = struct.unpack('d', double_num)
    return double_num[0]


def bytes_to_uint32(listBytes, startIndex):
    four_bytes = listBytes[startIndex:startIndex + 4]
    uint32_num = struct.pack('4B', *four_bytes)
    uint32_num = struct.unpack('I', uint32_num)
    return uint32_num[0]


class ANPacket(object):
    """docstring for ANPacket"""
    PACKET_ID_ACKNOWLEDGE = 0
    PACKET_ID_REQUEST = 1
    PACKET_ID_BOOT_MODE = 2
    PACKET_ID_DEVICE_INFORMATION = 3
    PACKET_ID_RESTORE_FACTORY_SETTINGS = 4
    PACKET_ID_RESET = 5
    PACKET_ID_PRINT = 6
    PACKET_ID_FILE_TRANSFER_REQUEST = 7
    PACKET_ID_FILE_TRANSFER_ACKNOWLEDGE = 8
    PACKET_ID_FILE_TRANSFER = 9

    PACKET_ID_SYSTEM_STATE = 20
    PACKET_ID_RAW_SENSORS = 28

    def __init__(self, length, id):
        super(ANPacket, self).__init__()
        self.length = length
        self.id = id
        self.data = [None] * self.length
```

```
class ANPacket20(object):
    """docstring for ANPacket20"""
    GNSS_NO_FIX = 0
    GNSS_2D_FIX = 1
    GNSS_3D_FIX = 2
    GNSS_SBAS_FIX = 3
    GNSS_DIFFERENTIAL_FIX = 4
    GNSS_OMNISTAR_FIX = 5
    GNSS_RTK_FLOAT_FIX = 6
    GNSS_RTK_FIXED_FIX = 7

    def __init__(self, packet):
        super(ANPacket20, self).__init__()
        # self.arg = arg

        self.statusAlert = False
        self.systemFailure = False
        self.accelerometerSensorFailure = False
        self.gyroscopeSensorFailure = False
        self.magnetometerSensorFailure = False
        self.pressureSensorFailure = False
        self.gnssFailure = False
        self.accelerometerOverRange = False
        self.gyroscopeOverRange = False

        self.magnetometerOverRange = False
        self.pressureOverRange = False
        self.minimumTemperatureAlarm = False
        self.maximumTemperatureAlarm = False
        self.dvlFailure = False
        self.ingressAlarm = False
        self.gnssAntennaAlarm = False
        self.dataOverflowAlarm = False

        self.orientationInitialised = False
        self.navigationInitialised = False
        self.headingInitialised = False
        self.utcTimeValid = False
        self.gnssFixType = self.GNSS_NO_FIX
        self.gnssFix = False
        self.event1Flag = False
        self.event2Flag = False
        self.internalGnssActive = False
        self.dvlTrackingActive = False
        self.bottomTrackingActive = False
        self.pressureDepthActive = False
        self.externalPositiveActive = False
        self.externalVelocityActive = False
        self.externalHeadingActive = False
```

```
        if packet[0] != 0:
            self.statusAlert = True
            if (packet[0] & 0x01) != 0:
                self.systemFailure = True
            if (packet[0] & 0x02) != 0:
                self.accelerometerSensorFailure = True
            if (packet[0] & 0x04) != 0:
                self.gyroscopeSensorFailure = True
            if (packet[0] & 0x08) != 0:
                self.magnetometerSensorFailure = True
            if (packet[0] & 0x10) != 0:
                self.pressureSensorFailure = True
            if (packet[0] & 0x20) != 0:
                self.gnssFailure = True
            if (packet[0] & 0x40) != 0:
                self.accelerometerOverRange = True
            if (packet[0] & 0x80) != 0:
                self.gyroscopeOverRange = True

        if packet[1] != 0:
            self.statusAlert = True
            if (packet[1] & 0x01) != 0:
                self.magnetometerOverRange = True
            if (packet[1] & 0x02) != 0:
                self.pressureOverRange = True
            if (packet[1] & 0x04) != 0:
                self.minimumTemperatureAlarm = True
            if (packet[1] & 0x08) != 0:
                self.maximumTemperatureAlarm = True
            if (packet[1] & 0x10) != 0:
                self.dvlFailure = True
            if (packet[1] & 0x20) != 0:
                self.ingressAlarm = True
            if (packet[1] & 0x40) != 0:
                self.gnssAntennaAlarm = True
            if (packet[1] & 0x80) != 0:
                self.dataOverflowAlarm = True

        if (packet[2] & 0x01) != 0:
            self.orientationInitialised = True
        if (packet[2] & 0x02) != 0:
            self.navigationInitialised = True
        if (packet[2] & 0x04) != 0:
            self.headingInitialised = True
        if (packet[2] & 0x08) != 0:
            self.utcTimeValid = True

        self.gnssFixType = (packet[2] & 0x70) >> 4
```

```
        if self.gnssFixType > self.GNSS_NO_FIX:
            self.gnssFix = True
        if (packet[2] & 0x80) != 0:
            self.event1Flag = True
        if (packet[3] & 0x01) != 0:
            self.event2Flag = True
        if (packet[3] & 0x02) != 0:
            self.internalGnssActive = True
        if (packet[3] & 0x04) != 0:
            self.dvlTrackingActive = True
        if (packet[3] & 0x08) != 0:
            self.bottomTrackingActive = True
        if (packet[3] & 0x10) != 0:
            self.pressureDepthActive = True
        if (packet[3] & 0x20) != 0:
            self.externalPositiveActive = True
        if (packet[3] & 0x40) != 0:
            self.externalVelocityActive = True
        if (packet[3] & 0x80) != 0:
            self.externalHeadingActive = True

        self.utc_time = self.bytes_to_UINT32(packet, 4)
        self.microseconds = self.bytes_to_UINT32(packet, 8)
        # self.time = new DateTime(utc_time * 10000000 +
microseconds * 10)

        self.position = []
        self.position.append(self.bytes_to_double(packet,
12))
        self.position.append(self.bytes to double(packet,
20))
        self.position.append(self.bytes to double(packet,
28))

        self.velocity = []
        self.velocity.append(self.bytes to float(packet,
36))
        self.velocity.append(self.bytes_to_float(packet,
40))
        self.velocity.append(self.bytes_to_float(packet,
44))

        self.acceleration = []
        self.acceleration.append(self.bytes_to_float(packet,
48))
        self.acceleration.append(self.bytes_to_float(packet,
52))
        self.acceleration.append(self.bytes to float(packet,
56))
```

```
        self.gForce = self.bytes_to_float(packet, 60)

        self.orientation = []
        self.orientation.append(self.bytes to float(packet,
64))
        self.orientation.append(self.bytes_to_float(packet,
68))
        self.orientation.append(self.bytes_to_float(packet,
72))

        self.angularVelocity = []

self.angularVelocity.append(self.bytes_to_float(packet, 76))

self.angularVelocity.append(self.bytes to float(packet, 80))

self.angularVelocity.append(self.bytes_to_float(packet, 84))

        self.positionStandardDeviation = []

self.positionStandardDeviation.append(self.bytes to float(pa
cket, 88))

self.positionStandardDeviation.append(self.bytes to float(pa
cket, 92))

self.positionStandardDeviation.append(self.bytes_to_float(pa
cket, 96))
        self.depth = self.bytes_to_float(packet, 100)

    def bytes_to_float(self, listBytes, startIndex):
        four bytes = listBytes[startIndex:startIndex + 4]
        float_num = struct.pack('4B', *four_bytes)
        float_num = struct.unpack('f', float_num)
        return float_num[0]

    def bytes_to_double(self, listBytes, startIndex):
        eight_bytes = listBytes[startIndex:startIndex + 8]
        double_num = struct.pack('8B', *eight_bytes)
        double_num = struct.unpack('d', double_num)
        return double_num[0]

    def bytes_to_UINT32(self, listBytes, startIndex):
        four_bytes = listBytes[startIndex:startIndex + 4]
        uint32_num = struct.pack('4B', *four_bytes)
        uint32_num = struct.unpack('I', uint32_num)
        return uint32_num[0]
```

```
class ANPacket28(object):
    """docstring for ANPacket28"""

    def __init__(self, packet):
        super(ANPacket28, self).__init__()
        # self.ANPacket = ANPacket
        self.accelerometers = []
        # self.accelerometers[0] = 0

self.accelerometers.append(self.bytes_to_float(packet, 0))
        # self.accelerometers[1] = 0

self.accelerometers.append(self.bytes_to_float(packet, 4))
        # self.accelerometers[2] = 0

self.accelerometers.append(self.bytes_to_float(packet, 8))
        self.gyroscopes = []
        # self.gyroscopes[0] = 0
        self.gyroscopes.append(self.bytes to float(packet,
12))
        # self.gyroscopes[1] = 0
        self.gyroscopes.append(self.bytes_to_float(packet,
16))
        # self.gyroscopes[2] = 0
        self.gyroscopes.append(self.bytes to float(packet,
20))
        self.magnetometers = []
        # self.magnetometers[0] = 0

self.magnetometers.append(self.bytes to float(packet, 24))
        # self.magnetometers[1] = 0

self.magnetometers.append(self.bytes_to_float(packet, 28))
        # self.magnetometers[2] = 0

self.magnetometers.append(self.bytes to float(packet, 32))
        self.imuTemperature = self.bytes to float(packet,
36)
        self.pressure = self.bytes_to_float(packet, 40)
        self.pressureTemperature =
self.bytes to float(packet, 44)

    def bytes_to_float(self, listBytes, startIndex):
        four_bytes = listBytes[startIndex:startIndex + 4]
        float_num = struct.pack('4B', *four_bytes)
        float_num = struct.unpack('f', float_num)
        return float_num[0]
```

```
class ANPacket36(object):
    """docstring for ANPacket28"""

    def __init__(self, packet):
        super(ANPacket36, self).__init__()
        self.bodyVelocityX = bytes_to_float(packet, 0)
        self.bodyVelocityY = bytes_to_float(packet, 4)
        self.bodyVelocityZ = bytes_to_float(packet, 8)


class ANPacket37(object):
    """docstring for ANPacket28"""

    def __init__(self, packet):
        super(ANPacket37, self).  init  ()
        self.earthAccelerationX = bytes_to_float(packet, 0)
        self.earthAccelerationY = bytes_to_float(packet, 4)
        self.earthAccelerationZ = bytes to float(packet, 8)


class ANPacket39(object):
    """docstring for ANPacket28"""

    def __init__(self, packet):
        super(ANPacket39, self).  init  ()
        self.roll = bytes to float(packet, 0)
        self.pitch = bytes_to_float(packet, 4)
        self.heading = bytes_to_float(packet, 8)


class ANPacketDecoder(object):
    """docstring for ANPacketDecoder"""
    HEADER_SIZE = 5
    MAXIMUM_SIZE = 255
    BUFFER SIZE = 8 * (MAXIMUM SIZE + HEADER SIZE)
    crc16Table = [0x0000, 0x1021, 0x2042, 0x3063, 0x4084,
0x50a5, 0x60c6, 0x70e7, 0x8108, 0x9129, 0xa14a, 0xb16b,
                  0xc18c, 0xd1ad, 0xe1ce, 0xf1ef, 0x1231,
0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
                  0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd,
0xc39c, 0xf3ff, 0xe3de, 0x2462, 0x3443, 0x0420, 0x1401,
                  0x64e6, 0x74c7, 0x44a4, 0x5485, 0xa56a,
0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
                  0x3653, 0x2672, 0x1611, 0x0630, 0x76d7,
0x66f6, 0x5695, 0x46b4, 0xb75b, 0xa77a, 0x9719, 0x8738,
                  0xf7df, 0xe7fe, 0xd79d, 0xc7bc, 0x48c4,
0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
                  0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948,
0x9969, 0xa90a, 0xb92b, 0x5af5, 0x4ad4, 0x7ab7, 0x6a96,
```

```
                    0x1a71, 0x0a50, 0x3a33, 0x2a12, 0xdbfd,
0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
                    0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22,
0x3c03, 0x0c60, 0x1c41, 0xedae, 0xfd8f, 0xcdec, 0xddcd,
                    0xad2a, 0xbd0b, 0x8d68, 0x9d49, 0x7e97,
0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
                    0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b,
0xaf3a, 0x9f59, 0x8f78, 0x9188, 0x81a9, 0xb1ca, 0xa1eb,
                    0xd10c, 0xc12d, 0xf14e, 0xe16f, 0x1080,
0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
                    0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d,
0xd31c, 0xe37f, 0xf35e, 0x02b1, 0x1290, 0x22f3, 0x32d2,
                    0x4235, 0x5214, 0x6277, 0x7256, 0xb5ea,
0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
                    0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466,
0x6447, 0x5424, 0x4405, 0xa7db, 0xb7fa, 0x8799, 0x97b8,
                    0xe75f, 0xf77e, 0xc71d, 0xd73c, 0x26d3,
0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
                    0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8,
0x89e9, 0xb98a, 0xa9ab, 0x5844, 0x4865, 0x7806, 0x6827,
                    0x18c0, 0x08e1, 0x3882, 0x28a3, 0xcb7d,
0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
                    0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1,
0x1ad0, 0x2ab3, 0x3a92, 0xfd2e, 0xed0f, 0xdd6c, 0xcd4d,
                    0xbdaa, 0xad8b, 0x9de8, 0x8dc9, 0x7c26,
0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
                    0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b,
0xbfba, 0x8fd9, 0x9ff8, 0x6e17, 0x7e36, 0x4e55, 0x5e74,
                    0x2e93, 0x3eb2, 0x0ed1, 0x1ef0]

    def __init__(self):
        super(ANPacketDecoder, self).  init  ()
        # self.arg = arg
        self.buffer = [None] * self.BUFFER_SIZE
        self.bufferLength = 0

    def calculateHeaderLRC(self, data, offset):
        return self.to_byte((((data[offset + 0] +
data[offset + 1] + data[offset + 2] + data[offset + 3]) ^
0xFF) + 1))

    def calculateCRC16(self, data, offset, length):
        crc = 0xFFFF
        for i in range(offset, offset + length):
            crc = ((crc << 8) ^ self.crc16Table[((crc >> 8)
^ data[i]) & 0xFF]) & 0xFFFF
        return crc

    def stokeBuffer(self, data, offset, length):
```

```
        if self.bufferLength + length > self.BUFFER_SIZE:
            length = self.BUFFER_SIZE - self.bufferLength
        for i in range(0, length):
            self.buffer[self.bufferLength] = data[offset +
i]
            self.bufferLength += 1

    def packetDecode(self):
        packet = None
        decodeIterator = 0
        while decodeIterator + self.HEADER_SIZE <=
self.bufferLength:
            headerLRC = self.buffer[decodeIterator]
            decodeIterator += 1
            if headerLRC ==
self.calculateHeaderLRC(self.buffer, decodeIterator):
                id = self.buffer[decodeIterator]
                decodeIterator += 1
                length = self.buffer[decodeIterator]
                decodeIterator += 1
                crc = self.buffer[decodeIterator]
                decodeIterator += 1
                crc |= self.buffer[decodeIterator] << 8
                decodeIterator += 1
                if decodeIterator + length >
self.bufferLength:
                    decodeIterator -= 5
                    break
                if crc == self.calculateCRC16(self.buffer,
decodeIterator, length):
                    packet = ANPacket(length, id)
                    for i in range(0, length):
                        packet.data[i] =
self.buffer[decodeIterator + i]
                    decodeIterator += length
                    break
        if decodeIterator < self.bufferLength:
            if decodeIterator > 0:
                for i in range(0, self.bufferLength -
decodeIterator):
                    self.buffer[i] =
self.buffer[decodeIterator + i]
                self.bufferLength -= decodeIterator
        else:
            self.bufferLength = 0
        return packet

    def packetEncode(self, packet):
        # data.lengt = packet.length + Header_size
```

```
        # in c#  byte[] data = new byte[packet.length +
HEADER_SIZE];
        data = [None] * 5
        crc = self.calculateCRC16(packet.data, 0,
packet.length)
        data[1] = self.to_byte(packet.id)
        data[2] = self.to_byte(packet.length)
        data[3] = self.to_byte(crc & 0xFF)
        data[4] = self.to_byte((crc >> 8) & 0xFF)
        data[0] = self.calculateHeaderLRC(data, 1)
        for i in range(0, packet.length):
            data.append(packet.data[i])
        return data

    def to_byte(self, int_val):
        return ctypes.c_ubyte(int_val).value
```

B-1 Spesifikasi Navigasi Sublocus

Parameter	**Nilai**
Akurasi Posisi Horisontal (GPS)	0.8 m
Parameter	**Nilai**
Akurasi Posisi Horisontal (DVL)	0.08%
Akurasi Posisi Horisontal (Log)	0.4%
Akurasi Kedalaman	0.4m
Parameter	**Nilai**
Akurasi Roll dan Pitch	0.01
Akurasi Heading/Yaw	0.05
Rentang Orientasi	Tak terbatas
Waktu Panas	2 detik
Waktu North Seeking	< 60 detik
Filter Tingkat Internal	1000 Hz
Nilai Output Data	>1000 Hz

B-2 Spesifikasi Sensor Sublocus

Parameter	**Akselerometer**	**Giroskop**	**Magnetometer**	**Tekanan**
Jarak	10g	490	8 G	400 bar
Kerapatan Kebisigan	300 ug	0.0002 s	210 Ug	-
Ketidaklineran	< 0.03%	< 0.005 %	< 0.05%	-
Stabilitas Bias	50 ug	0.05	-	-
Stabilitas Skala	< 0.06 %	< 0.02%	<0.05%	-

Kesalahan Alignment Cross-Axis	< 0.05	< 0.02	0.05	-
Bandwidth	200 Hz	440 Hz	110 Hz	10 Hz

B-2 Spesifikasi GPS Sublocus

Parameter	Nilai
Sistem Navigasi yang Didukung	GPS L1 GLONASS L1 GALILEO E1 BeiDou B1
Sistem SBAS yang Didukung	WAAS EGNOS MSAS GAGAN QZSS
Update Rate	10 Hz
Hot Start First Fix	3 s
Cold Start First Fix	30 s
Akurasi Posisi Horisontal	1.2 m
Akurasi Posisi Horisontal	0.5 m

B-3 Spesifikasi Komunikasi Sublocus

Parameter	Nilai
Antarmuka	RS232 atau RS422
Kecepatan	1200 sampai 10M baud
Protokol	AN Packet Protocol
Periferal Antarmuka	2x GPIO and 1X Auxiliary RS232
Level GPIO	5V atau RS232

B-4 Spesifikasi Perangkat Keras Sublocus

Parameter	Nilai
Rentang Kedalaman	3000m
Tegangan Operasi	18 sampai 50 V
Proteksi Masukan	100 V
Konsumsi Daya	6W
Kapasitas Baterai	>24 jam
Waktu Pengisian Baterai	30 menit
Daya Tahan Baterai	>10 tahun
Suhu Operasi	-40 sampai 85 C
MTBF	>50.000 jam

Batas Kejut	25 g
Dimensi	170x170x133 mm
Berat di Udara	6.4 kg
Berat di Air	3.9 kg

B-5 Spesifikasi Elektronika Sublocus

Parameter	Minimal	Typical	Maksimum
Power Supply			
Tegangan Input Supply	18 V		50 V
Rentang Masukan Proteksi	-100 V		100 V
RS232			
Tx Tegangan Rendah		-5.4 V	-5 V
Tx Tengangan Tinggi	5 V	5.4 V	
Tx Arus Rangkaian Pendek			60 Ma
Rx Ambang Batas Rendah	0.6 V	1.2 V	
Rx Ambang Batas Tinggi		1.5 V	2.0 V
RS422			
Tx Keluaran Diferensial		1.5 V	
Parameter	**Minimal**	**Typical**	**Maksimum**
Tx Arus Rangkaian Pendek			250 Ma
Rx Diferensial	-0.2 V		-0.05 V
GPIO			
Tegangan Keluaran Rendah	0 V		0.3 V
Tegangan Keluaran Tinggi	4.8 V		5 V
Teganan Masukan	-20 V		20 V
Masukan Ambang Batas Rendah			1.5 V
Masukan Ambang Batas Tinggi	3.5 V		
Arus Keluaran			5 Ma
GPS Antena			
Tegangan Pasokan Antena Aktif	2.65 V		2.85 V
Antena Pasokan Saat ini			54 Ma

B-6 Kinerja Antena GNSS

Parameter	Nilai
Sistem Navigasi yang Didukung	GPS L1/L2/L5 GLONASS G1/G2/G3 GALILIEO E1/E5 BeiDou B1/B2 L-Band Corrections
Sistem SBAS yang Didukung	WAAS EGNOS

	MSAS GAGAN QZSS
Elemen Antena Gain	4dBiC
Polarisasi	Polarisasi Tangan Kanan
LNA Gain	28 dB
Kebisingan	<2 dB
Rentang Tegangan Operasional	2.5 – 16 V DC
Konsumsi Arus	20 mA, maksimum 25 mA
Perlindungan ESD	15KV

B-7 Perangkat Keras

Parameter	Nilai
Dimensi	Diameter 67mm x 23 mm
Panjang Kabel	3m
Temperature Operasional	-40 °C hingga 85 °C
Berat	320 gram
Bahan Dasar	316 stainless steel
RoHS Compliant	Ya
Shock	Vertical 50 G, Axis 30 G
Rentang Tekanan Maksimal	300 bar (3000 meter)

B-8 Kabel Koaksial

Parameter	Nilai
Bahan	Polyether Polyurethane 4350
Warna	Oranye Visibilitas Tinggi
Suhu Operasional	40 °C to 85 °C
Impedansi	50 Ohm
Minimum Bend Radius	15 mm

B-9 Spesifikasi Underwater Monitor

Parameter	Nilai
Ukuran Layar Diagonal	10”
Rasio Tampilan Panel	4:3
Resolusi	1024x768
Sinyal Masukan	PC (VGA)
Format HDMI	1080p60; 1080p50; 1080i60; 1080i50
Backlight	LED
Angle View	50/70 (U / D), 70/70 L / R)
Kecerahan	250 cd/m2
Rasio Kontras	600:1

Tegangan Masukan	DC9~ 15 V
Konsumsi Daya	<7W
Suhu Kerja	-10°C ~ 50°C
Suhu Penyimpanan	-10°C ~ 70°C
Tingkat Kedalaman	60 m
Berat	2.4 kg/ 5.29 lbs
Ukuran	232mm x 183mm x 57mm

B-10 Spesifikasi RS422

Panjang Kabel	1 meter
Chip	FTDI
Kecepatan Transfer Data	3 Mbps
Konektor Device Port	DB-9/M x 1
Konektor Host Port	USB Tipe A/M
Lapisan Penutup	PVC
LED	3
Mode Daya	BUS
USB Versi Compliant	USB 1.1

B-11 Spesifikasi USB to RS422 *Converter*

Panjang Kabel	1 meter
Chip	FTDI
Kecepatan Transfer Data	3 Mbps
Konektor Device Port	Terminal Block x 4
Konektor Host Port	USB Tipe A/M
Lapisan Penutup	Plastik
Mode Daya	BUS
USB Versi Compliant	USB 1.1

B-12 Tipe Data ANPP

Abbreviation	Bytes	Also known as
u8	1	unsigned char, unsigned byte, uint8_t
s8	1	char, byte, int8_t
u16	2	unsigned short, uint16_t
s16	2	short, int16_t
u32	4	unsigned int, unsigned long, uint32_t
s32	4	int, long, int32_t
u64	8	unsigned long long, uint64_t
s64	8	long long, int64_t
fp32	4	float
fp64	8	double

B-13 ANPP Packet 20

System State Packet				
Packet ID				20
Length				112
Field #	**Bytes Offset**	**Data Type**	**Size**	**Description**
1	0	u16	2	System status, see section 9.8.1.1
2	2	u16	2	Filter status, see section 9.8.1.2
3	4	u32	4	Unix time seconds, see section 9.8.1.4
4	8	u32	4	Microseconds, see section 9.8.1.5
5	12	fp64	8	Latitude (rad)
6	20	fp64	8	Longitude (rad)
7	28	fp64	8	WGS84 Height (m)
8	36	fp32	4	Velocity north (m/s)
9	40	fp32	4	Velocity east (m/s)
10	44	fp32	4	Velocity down (m/s)
11	48	fp32	4	Body acceleration X (m/s/s)
12	52	fp32	4	Body acceleration Y (m/s/s)
13	56	fp32	4	Body acceleration Z (m/s/s)
14	60	fp32	4	G force (g)
15	64	fp32	4	Roll (radians)
16	68	fp32	4	Pitch (radians)
17	72	fp32	4	Heading (radians)
18	76	fp32	4	Angular velocity X (rad/s)
19	80	fp32	4	Angular velocity Y (rad/s)
20	84	fp32	4	Angular velocity Z (rad/s)
21	88	fp32	4	Latitude standard deviation (m)
22	92	fp32	4	Longitude standard deviation (m)
23	96	fp32	4	Height standard deviation (m)
24	100	fp32	4	Depth (m)
25	104	fp32	4	Altitude (m), see section 9.8.1.6
26	108	fp32	4	Heave (m)

B-14 ANPP Packet 28

Raw Sensors Packet				
Packet ID				28
Length				48
Field #	**Bytes Offset**	**Data Type**	**Size**	**Description**
1	0	fp32	4	Accelerometer X (m/s/s)
2	4	fp32	4	Accelerometer Y (m/s/s)
3	8	fp32	4	Accelerometer Z (m/s/s)
4	12	fp32	4	Gyroscope X (rad/s)
5	16	fp32	4	Gyroscope Y (rad/s)
6	20	fp32	4	Gyroscope Z (rad/s)
7	24	fp32	4	Magnetometer X (mG)
8	28	fp32	4	Magnetometer Y (mG)
9	32	fp32	4	Magnetometer Z (mG)
10	36	fp32	4	IMU Temperature (deg C)
11	40	fp32	4	Pressure (Pascals)
12	44	fp32	4	Pressure Temperature (deg C)

RIWAYAT HIDUP

Amirotul Khoiro lahir di Surabaya pada tanggal 23 Juli 1996. Penulis adalah anak kedua dari dua bersaudara. Penulis menempuh pendidikan dasar di SDN Semampir I Sidoarjo pada tahun 2003-2009, kemudian SMPN 1 Waru Sidoarjo tahun 2009-2012, dan lulus dari SMA Negeri 3 Sidoarjo pada tahun 2015. Pada tahun 2015 juga, penulis diterima sebagai mahasiswa di jurusan D3 Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS). Semasa kuliah, penulis aktif sebagai staf Departemen HMKI HIMAD3TEKTRO 2016-2017 dan staf Departemen Kominfo BEM FTI ITS 2016-2017. Selain kegiatan di kampus, penulis juga aktif sebagai Atlet Judo Puslatkab Sidoarjo. Sejak 2016, penulis juga aktif sebagai anggota ACSL (Automation Computer System Laboratory) ITS. Penulis telah menyelesaikan magang dan pengerjaan Tugas Akhir ini di PT Bhimasena Research and Development.

E-mail: amirotulkhoiro@gmail.com

Halaman ini sengaja dikosongkan